What to Write and How to Write It

Modeling Content Selection and Generation for Text Summarization

Thesis Proposal

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

December 10th, 2018

Abstract

Automatic text summarization is a long standing NLP problem that has recently seen an uptick in interest due to flexible, high capacity deep learning based sequence-to-sequence transduction models. While many researchers are turning to complex neural networks to perform end-to-end summary generation, we argue that this unnecessarily obscures the underlying sub-tasks for summarization. Instead, we propose an alternative research agenda, focusing on two key summarization subtasks: *i)* estimating the general importance of text content for summary inclusion, and *ii)* generating text that is faithful, a notion we formally define, to said important content. With respect to problem *i)* we propose several novel methods for working in both low context, streaming news scenarios, as well as, standard single document summarization settings, including feature-based and deep learning models of content importance. On the latter problem, we study the utility of using extractive summarization algorithms as a preprocessing stage for a sequence-to-sequence based abstractive summarizer, and finally introduce a novel algorithm for generating text that is faithful, i.e. respects prior beliefs about structured knowledge in a database, in an effort to provide stronger guarantees about summary reliability.

Contents

1 Introduction **1**

1.1 Contributions . . . 2

2 Related Work **3**

2.1 Faithful Generation . . . 3

3 Feature-Based Models of Sentence Salience **3**

3.1 Stream Summarization Task . . . 4

3.2 Features for Sentence Salience Estimation . . . 6

3.3 Model 1: Salience-Biased Affinity Propagation Clustering . . . 9

3.3.1 Salience Estimation . . . 9

3.3.2 Salience-biased Affinity Propagation . . . 10

3.3.3 Data . . . 11

3.3.4 Experiments . . . 11

3.3.5 Results . . . 12

3.4 Model 2: Learning-to-Summarize . . . 13

3.4.1 Oracle Policy and Loss Function . . . 15

3.4.2 Salience Estimation . . . 16

3.4.3 Data . . . 16

3.4.4 Experiments . . . 17

3.4.5 Results . . . 17

4 Deep Learning Models of Content Salience **19**

4.1 Deep Learning Models of Sentence Salience . . . 20

4.1.1 Sentence Encoders . . . 20

4.1.2 Sentence Extractors . . . 21

4.1.3 Data . . . 24

4.1.4 Experiments . . . 25

4.1.5 Results . . . 26

4.2 Word Importance Estimation in Deep Learning Models . . . 29

4.2.1 Proposed Model . . . 30

5 Faithful Text Generation **32**

5.1 Related Work . . . 33

5.2 Structured Data Model . . . 34

5.3 Text Data Model . . . 35

5.4 Applications and Data . . . 36

6 Research Plan **37**

7 Conclusion **37**

List of General Definitions

$\mathbb{R}$	The set of real numbers.
$\mathbb{N}$	The set of integers.
$\mathbb{N}^+$	The set of positive integers.
$\mathcal{X}$	A generic discrete space.
$\lvert x\rvert$	Cardinality of a discrete set or sequence x.
$\mathbb{1}\{\cdot\}$	Iverson bracket, i.e. evaluates to 1 if proposition in brackets evaluates to True.
$\llbracket n \rrbracket$	The index set $\{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}^+$.
$\mathbb{E}[\cdot]$	The expected value. When ambiguous we will use a subscript $\mathbb{E}_p[\cdot]$ to specify the distribution p.
H	The Shannon entropy, $-\sum_{x\in\mathcal{X}} p(x) \log p(x)$. We will use the subscript $H_{p(\cdot)}$ to specify the
$\mathcal{D}$	A dataset.
$\mathcal{L}$	An objective function.

List of Chapter 3 Definitions

q An event query string, e.g. “Hurricane Sandy”.

$\mathcal{S}(q)$ A sentence stream, i.e. a time-ordered sequence of sentences for a particular query q. Typically we will omit the the query as it will be clear from the context which stream we are refferring to.

s A sentence from the sentence stream $\mathcal{S}$.

ϕ A function $\phi : \mathcal{S} \to \mathbb{R}^d$ mapping sentences to a d-dimensional feature vector.

$\mathcal{N}(q)$ The set of nuggets associated with query q. Typically we will omit the the query as it will be clear from the context which nuggets we are refferring to.

n An event nugget text.

$\mathcal{U}$ A distinguished set of sentences $\{s_i\} \subseteq \mathcal{S}$ that were emitted by a stream summarization algorithm, i.e. the extract summary for a stream $\mathcal{S}(q)$.

$\hat{\mathcal{N}}$ The set of nuggets contained by the predicted extract summary $\mathcal{U}$, i.e. $\big\{n \in \mathcal{N} \mid (\exists s \in \mathcal{U})[n \in s]\big\}$.

y The salience of a sentence s. Salience scores are real valued.

$\mathbf{y}$ A vector of k salience values, i.e. $\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{R}^k$.

$\hat{y}$ The predicted salience of a sentence s. Salience scores are real valued.

$\hat{\mathbf{y}}$ A vector of k predicted salience values, i.e. $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_k) \in \mathbb{R}^k$.

$\mathcal{E}$ The set of exemplars selected by the Affinity Propagation algorithm.

1 Introduction

Everyday we ask friends and colleagues to do it, to tell us about books, newspaper articles, and other complex pieces of text, and, without more than a moment's reflection, they are able to compress their experience into succint statements in natural language that adequately sum up the thing being described. The ease with which we summarize belies the difficulty in writing programs that do so automatically, and so it would appear that the robust ability to create summaries is, as of 2018, a uniquely human capability. This partially explains its 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 long and varied as some of the more foundational tasks, e.g. parsing and tagging.

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***.

The completed and proposed methods of salience estimation cover a range of techniques including feature-based regression, clustering, learning-to-search, and deep learning. 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 representitive 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 in 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 noteable 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 section 3 and section 4.

In section 3 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 to the user, and timely – 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 4 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 data, and only in the news genre. This is similarly straightforward, given a set of related news articles (typically ten), select a subset of the sentences to use as the summary of the document set.

Salience estimation can also be used to guide ***abstractive summarization*** techniques, i.e. methods that produce novel summary text using a generative model. These types of models must (at the very least) implicitly select important content for conditioning their generative process. We propose a salience estimation component to explicitly model this selection process in addition to a generative model training regime which we briefly describe presently. (NOTE-TO-SELF: reorder this transition you dumb dumb.)

Research in abstractive summarization is increasing, in part due to the success of general purpose sequence-to-sequence transduction models in machine translation (MT) that have been ported to the summarization task. The ability of abstractive models to generate relatively fluent and meaningful summaries is impressive. However, they are also especially prone to generating generic statements that are not grounded by evidence in the input. This seems to be a well known problem among researchers working in generation, but it has not received much attention in the literature. 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 possesed by the model apriori. 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 structured data); the second player, the recognizer, evaluates the plausibility of the utterance with respect to the evidence. This overall regime will consitute our method of faithful generation, to be outlined in more detail in section 5. We propose evaluation both on an SDS task, as well as, two data-to-text generation test beds (Lebret et al., 2016; Wiseman 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, as well as a thorough evaluation of the linguistic and structural features critical to learning in the SDS task across a variety of genres.

3. Adaptation of the deep learning based SDS models to a news MDS task.

4. 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.

5. A novel method of combining salience estimation based extractive summarization with abstractive generation in the faithful generation paradigm.

Together, these contributions provide a recipe book 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 we describe related work (Section 2). Section 3 describes completed work on the feature-based approaches to sentence salience estimation. In section 4, we describe completed and ongoing work on deep learning models of salience estimation. We describe our planned approaches to faithful generation in section 5. Section 6 describes our research plan, before we finally conclude.

2 Related Work

(REWRITING THIS NOW)

2.1 Faithful Generation

In our proposed faithful generation framework, we use one model, the generator, to create a text utterance, e.g. a summary of a document or a description of a data table, and another model, the recognizer, to answer questions about the underlying document or table, using the utterance as evidence. A good utterance, then, is one that allows the recognizer on average to correctly answer those questions.

The conceptual underpinnings of faithful generation trace back to two ideas in the NLP literature. The first is round-tripping in MT evaluation (Somers, 2005; Rapp, 2009), i.e. a good summarization system should be able to translate a source language text to a target language text and then back to the source language with minimal corrpution between the original source and the back-translated source. 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. Similarly, minimum error rate training (MERT) (Och, 2003) and minimum Bayes-risk decoding (Kumar and Byrne, 2004) rerank n-best translations in order to directly optimize an evaluation metric like BLEU (Papineni et al., 2002).

Faithiful generation can be recast in the MERT framework by considering the evaluation metric to be the recognizer's classification accuracy on table data or cloze accuracy on text data. This is perhaps most similar to

3 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 coverage on the test set. 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 and is blissfully unaware of any lexical features. In this section we describe our features in the context of a stream summarization task, but the features are themselves fairly generic and likely obtainable 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, from a trained salience regression model, 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 in detail, before moving on to the sentence features, and finally the models and their evaluations.

3.1 Stream Summarization Task

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).

The scenario proposed by TREC was disaster summarization, where a participant model 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

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

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

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 $\mathbb{E}\left[\textsc{Gain}\right]$, and comprehensiveness, or Comp., which are precision and recall like measures of nugget coverage respectively. Let $\mathcal{U}$ be the set of sentences extracted by the summarization model for a query and let $\mathcal{N}$ be the set of nuggets for that same query. The set of nuggets recovered by the model is then $\hat{\mathcal{N}} = \{n \in \mathcal{N} \mid (\exists s \in \mathcal{U})[n \in s]\}$. The expected gain and comprehensiveness are then defined as

$$\mathbb{E}\left[\textsc{Gain}\right] = \frac{|\hat{\mathcal{N}}|}{|\mathcal{U}|} \quad \text{and} \quad \textsc{Comp.} = \frac{|\hat{\mathcal{N}}|}{|\mathcal{N}|}$$

respectively where $|\cdot|$ is the size of a set. An $\mathbb{E}\left[\textsc{Gain}\right]$ of 1 would mean that every sentence in the update summary contained a novel nugget. Comp. is the nugget recall, a score of 1 indicating that all nuggets were found in the update summary.

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.

In most of our experiments, we lack annotations about what nuggets if any are contained in a given sentence so we approximate the manual evaluation of the official TS track by automatically mapping sentences to nuggets if their semantic similarity (Guo and Diab, 2012) is a above a threshold. We report results across a sweep of threshold values from zero to one, with values closer to one a more conservative estimate of performance. We also concatenate the nugget texts to create a reference summary and report Rouge scores (Lin, 2004) for the update summary produced by the model.

If we had clairvoyant knowledge of which nuggets were contained in each sentence in the document stream, this task would be trivial in that the greedy algorithm in Figure 1 would return an approximately optimal summary. Since we do not know this information at test time, we attempt to approximate the if-statement in line 5 of the oracle by answering the following proxy questions:

1. How salient is sentence s with respect to query q?
2. How similar is the sentence s with respect to the set of previously selected updates $\hat{s} \in \mathcal{U}$?

```
procedure OracleSummarize
    $\mathcal{U} \leftarrow \varnothing$                                  ▷ Init. update summary.
    $\hat{\mathcal{N}} \leftarrow \varnothing$                                  ▷ Init. found nuggets.
    for $s \in \mathcal{S}(q)$ do
        if $\exists n$ s.t. $n \in s \wedge n \notin \hat{\mathcal{N}}$ then
            $\mathcal{U} \leftarrow \mathcal{U} \cup \{s\}$
            $\hat{\mathcal{N}} \leftarrow \hat{\mathcal{N}} \cup \{n \in s\}$
        end if
    end for
    return $\mathcal{U}$
end procedure
```

Figure 1: Clairvoyant oracle stream summarization algorithm. $\mathcal{S}(q) = (s_1, \ldots, s_T)$ is a time-ordered sequence of T sentences from the relevant document stream for query q.

As a rule we should only extract sentences that are relevant to the query and not too similar to sentences have previously extracted.

The SAP model uses regression to estimate the salience and then relies on a clustering algorithm to consider a non-redundant set of extract candidates. The LS model directly integrates redundancy as a feature in a salience model, in effect, answering the salience and similarity question simultaneously.

In the next subsection, we describe the different feature groups before describing each of the models used to perform the streaming summarization task.

3.2 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 ngrams will be shifting throughout the document stream. Instead we compute several groups of sentence features that are specifically helpful for the query focused 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 name.

Simple Surface Features The document stream that we will be operating over is incredibly noisy, full of automatically extracted article text from raw web pages in dramatically different formatting. As a result, the underlying text is often full of wep page headers, headlines to other stories, and irrelevant link text. We use these simple shallow surface features to characterize "normal" newswire sentences. These features include sentence length, the average number of capitalized words, document position, sentence length in words, and the average number of named entities. Length and position features have been used previously in other learning based models of sentence salience (Kupiec et al., 1995; Conroy et al., 2001).

Query Features To ensure the focus of the update summaries, 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

Event Type	**Event Queries**
Storm	hurricane isaac, hurricane sandy, midwest derecho, typhoon bopha
Earthquake	guatemala earthquake
Meteor Impact	russia meteor
Accident	buenos aires train crash, pakistan factory fire
Riot	egyptian riots
Protest	bulgarian protests, egyptian protests
Hostages	in amenas hostage crisis
Shooting	colorado shooting, sikh temple shooting
Bombing	boston marathon bombing, hyderabad explosion
Conflict	konna battle

Table 2: TREC Temporal Summarization query event types with example queries.

expansion. Queries in the TS data are manually labeled with event type; see Table 2 for the list of query event types we used. For the *earthquake* query type, for example, we find the following terms: “quake”, “temblor”, “seism”, and “aftershock”.

Language Model Scores We rely on a pair of language models to assess the likelihood of observing a given sentence in the document stream. The first language model is intended to be a generic news model, trained on New York Times and Associated Press sections of the Gigaword corpus (Graff and Cieri, 2003). This model serves a similar purpose as the simple surface features in helping to identify newswire-like sentences that are likely to contain informative content. The second model is query type specific and intended to focus the summarizer on sentences from the same domain of the query. For each query type, we constructed an in-domain corpus of Wikipedia news articles belonging to categories and pages related to the event type. The corpus is semi-automatically constructed: first a set of high-level categories are selected and then all pages belonging to those initial categories or any sub-categories are automatically added. For example, the language model for the event type “earthquake” is estimated from Wikipedia pages under the category Category:Earthquakes. For the actual summarization system features, we use the average token log likelihood under each model as a feature. We use the SRILM toolkit (Stolke, 2002) to implement a 5-gram Kneser-Ney model (Kneser and Ney, 1995) for both the general and type specific language models.

Geographic Relevance (SAP) The queries in the TS data are all grounded to physical events in the world and we would like to exploit that. For example, while there are sadly many reports of bombings or explosions at any given time around the world, when summarizing the query “boston marathon bombing” we are only really interested in news from a relatively proscribed geographic area, in this case, Boston, Massachusets. Implementing a geographic relevance signal is quite difficult in practice since we do not have ground truth location data for each event query, and some events like hurricanes can be on the move. We opt instead to extract all location mentions (using a named entity tagger) and get the lattitude and longitude coordinates of each mention using the Bing Maps API[1]. Location mentions within the same hour are clustered, and the cluster centers are used as the canonical event locations for the query at that time. For each

[1] `https://www.microsoft.com/en-us/maps/documentation`

document we compute the median distance of all location mentions to the event location cluster. We also compute the distance of the first location mention to the event location cluster. Sentences from the document share these two distance features.

Temporal Relevance (SAP) As we track a query over time, the document stream will ebb and flow as the underlying event unfolds. With hurricanes for example, forecasters are usually watching a tropical storm as it grows and predicitng where it might make landfall. As a result, news about a major hurricane grows gradually, peaking after its main contact with land and gradually diminishing. An earthquake on the otherhand gives no warning, and there is a sudden spike in the query relevant document stream. We would like to take advantage of spiking or bursty activity in the document stream as a signal for our summarization system. We do this by maintaining inverse document frequency (IDF) values for each hour of the document stream. We create 24 average TF-IDF features for each sentence as if it had occurred in each of the previous 24 hours. I.e. we fix the observed sentence level term frequencies and rescale them by IDF values cached from each of the previous 24 hours; a single feature value is obtained for each hour by averaging the sentence's bag of rescaled TF-IDF values. Large changes in any of the individual features should be representative of a spike in the news.

Document Frequency (L2S) We also compute the hour-to-hour percent change in document frequency of the stream. This feature helps gauge breaking developments in an unfolding event. As this feature is also heavily affected by the daily news cycle (larger average document frequencies in the morning and evening) we compute the 0-mean/unit-variance of this feature using the training streams to find the mean and variance for each hour of the day.

Stream Language Models (L2S) We construct several simple unigram language models that we update with each new document that we see in the stream. Using these models we compute the sum, average, and maximum token likelihood of the non-stop words in the sentence. We compute similar quantities restricted to the person, location, and organization named entities.

Update Similarity (L2S) The average and maximum cosine similarity of the current sentence to all previous updates is computed under both the TF-IDF bag-of-words and latent vector representation (Guo and Diab, 2012). We also include indicator features for when the set of updates is empty (i.e. at the beginning of a run) and when either similarity is 0.

Nugget Probability (L2S) When developing the L2S model, we had access to a small collection of human sentence/nugget judgements from the 2014 Temporal Summarization evaluation. We used these judgements to train an ngram based classifier to predict whether the sentence contained a nugget, and we use the probability of a nugget given the sentence text as a feature.

Single Document Summarization Features (L2S) Even though we are processing the stream one sentence at a time in the L2S model, we still have knowledge of the document that a sentence came from and so we can compute features that take into account

```
procedure SapSummarize
    $\mathcal{U} \leftarrow \varnothing$    ▷ Initialize an empty update summary.
    for $\mathcal{S}_t \in \mathcal{S}(q)$ do
        $\hat{\mathbf{y}} \leftarrow \text{salience}(\mathcal{S}_t, q)$    ▷ Estimate sentence salience for each $s \in \mathcal{S}_t$.
        $\mathbf{K} \leftarrow \textsc{Similarity}(\mathcal{S}_t, \mathcal{S}_t)$    ▷ Compute similarity between each $s_i, s_j \in \mathcal{S}_t$.
        $\mathcal{E} \leftarrow \textsc{AffinityPropagation}(\hat{\mathbf{y}}, \mathbf{K})$    ▷ Get exemplar sentences $\mathcal{E}$ with SAP.
        for $s \in \mathcal{E}$ do    ▷ Filter low salience or high summary similarity exemplars.
            if $\hat{\mathbf{y}}_s > \lambda_1 \wedge \max_{u \in \mathcal{U}} \textsc{Similarity}(u, s) < \lambda_2$ then
                $\mathcal{U} \leftarrow \mathcal{U} \cup \{s\}$
            end if
        end for
    end for
    return $\mathcal{U}$
end procedure
```

Figure 2: Salience-biased Affinity Propagation (SAP) clustering based stream summarization algorithm. We process the stream in time ordered batches of sentences $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_T = \mathcal{S}(q)$ with each $\mathcal{S}_t = (s_1^{(t)}, \ldots, s_{|\mathcal{S}_t|}^{(t)})$ a time ordered sequence of sentences from the relevant document stream for query q.

the other sentences in the document. We used several unsupervised sentence ranking methods from the single document summarization literature to get several sentence rank features. Inspired be the SumBasic summarizer (Nenkova and Vanderwende, 2005), we compute the average and sum of unigram probability of each sentence using sentence's document unigram distribution. Inpsired by Guo et al. (2013), for each sentence we compute the average cosine distance to the remaining sentences in it's enclosing document. Finally, we compute the document level centroid rank (Radev et al., 2000) and LexRank (Erkan and Radev, 2004) for each sentence.

3.3 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 processesing 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 it's salience estimation is high relative to the batch.

Finally, given a set of exemplar sentences $\mathcal{E}$, we remove any sentences whose similarity to any previous update is above a threshold. The remaining exemplars are added to the update summary $\mathcal{U}$. The final algorithm is presented in Figure 2.

3.3.1 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 3.2 we create a seperate radial basis function (RBF) kernel

$$k_g(s, s') = \sigma_g^2 \exp\left(-\frac{||\phi_g(s) - \phi_g(s')||^2}{2\ell_g}\right)$$

where $\phi_g : \mathcal{S} \to \mathbb{R}^{d_g}$ maps sentences to a d_g-dimensional vector of features for group g and σ_g^2, ℓ_g are kernel parameters. The actual Gaussian process regressor is specified by the summation kernel $k(s, s') = \sum_g k_g(s, s')$ of the individual group kernels.

We fit a seperate regressor for each query q in our training set by sampling a set of 1000 sentences $\mathbf{s} = (s_1, \ldots, s_{1000})$ drawn uniformly from $\mathcal{S}(q)$ and computing their corresponding salience scores $\mathbf{y} = (y_1, \ldots, y_{1000})$. We fit the kernel parameters with gradient descent to maximize the likelihood of $\mathbf{y}$. At prediction time for a test query, we use the average prediction of each training query regressor to obtain a salience estimate $\hat{y}$ for each sentence in the test stream.

3.3.2 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.

Given a set of d datapoints $\mathcal{X} = \{x_1, \ldots, x_d\}$, AP finds a set of exemplar datapoints $\mathcal{E} \subset \mathcal{X}$ and an assignment $A : \mathcal{X} \to \mathcal{E}$ of the remaining points to one of the exemplars. Exemplars map to themselves, i.e. for $x \in \mathcal{E}$, $A(x) = x$. The configuration of exemplars and exemplar assignments is represented as a factor graph, with the following objective:

$$\mathcal{L}(\mathcal{E}, A) = \exp\left[\sum_{i=1}^{d} \text{affinity}\left(x_i, A(x_i)\right) + \log \delta_i\right]$$

	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 3: System ROUGE performance.

where $\delta_i = \begin{cases} 0 & \text{if } A(x_i) \neq x_i \text{ and } \exists j : j \neq i \wedge A(x_j) = x_i \\ 1 & \text{otherwise} \end{cases}$ is a constraint that enforces all clusters have one and only one exemplar and affinity : $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a function describing the pairwise similarity of points in $\mathcal{X}$. The max-product message passing algorithm, a form of loopy belief propagation, can be used to find a configuration that approximately maximizes the objective function (Dueck, 2009).

We can refactor the first term in the exponent to be

$$\sum_{x_i \in \mathcal{E}} \text{affinity}(x_i, x_i) + \sum_{x_j \in \mathcal{X} \setminus \mathcal{E}} \text{affinity}(x_j, A(x_j))$$

where the first term is the sum of the affinity factors of the exemplars to themselves, and the second term consists of pairwise factors between each non-exemplar datapoint and it's exemplar. In the typical case the exemplar self-affinity factors are simply set to a constant indicating all points are equally likely apriori to serve as exemplars. In our case, we have some prior beliefs about the importance of a given datapoint as expressed by our salience predictions $\hat{\mathbf{y}}_i$. Replacing the self-affinity potentials with our salience predictions, the pairwise potentials with our semantic similarity function from the previous section, and setting $\mathcal{X}$ to be a batch of sentences $\mathcal{S}_t \subset \mathcal{S}$, we arrive at the salience-biased affinity propagation objective:

$$\mathcal{L}(\mathcal{E}, A) = \exp \left[\sum_{s \in \mathcal{E}} \hat{\mathbf{y}}_s + \sum_{s \in \mathcal{S}_t \setminus \mathcal{E}} \textsc{Similarity}(s, A(s)) + \sum_{i=1}^{|\mathcal{S}_t|} \log \delta_i \right].$$

When we estimate a sentence to be more salient it is more likely apriori 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 it's likelihood of being assigned as an exemplar.

3.3.3 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 Temporal Summarization tracks. This data contained 25 events and their query strings, period of interest, 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.

3.3.4 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 F1 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 sentences salience using the average predictions of the 23 other models.

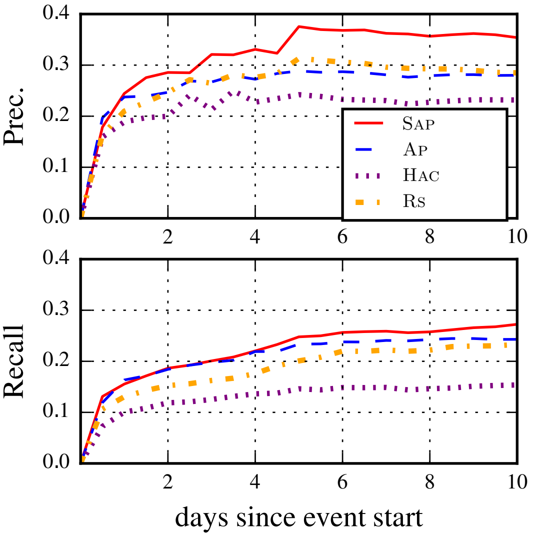

Figure 3: System ROUGE-1 performance over time.

Since we lacked gold judgements 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 it's 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 F1 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 F1 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 adding 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.

3.3.5 Results

Table 3 shows our results for system output samples against the full summary of nuggets using ROUGE. This improvement is statistically significant for all ngram precision, recall, and F1 scores at the $\alpha = .01$ level using the Wilcoxon signed-rank test. SAP maintains its performance above the baselines over time as well. Figure 3 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 4 shows the $\mathbb{E}$ [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 $\mathbb{E}$ [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.

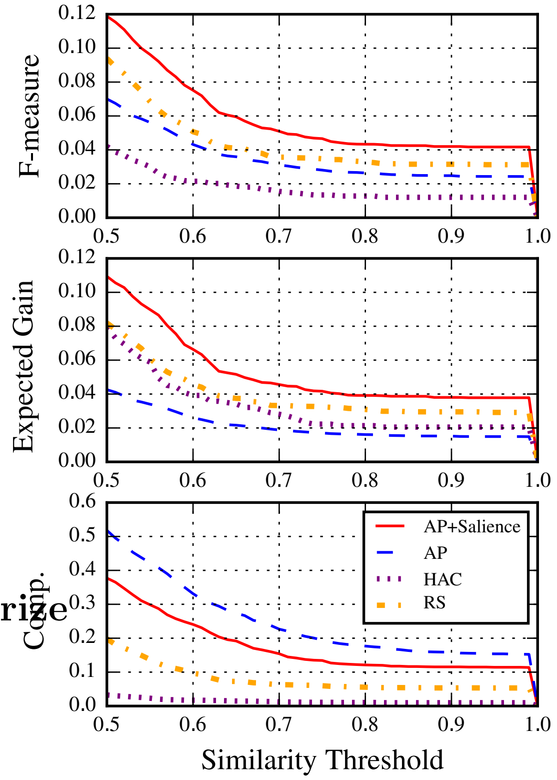

Figure 4: Expected Gain and Comprehensiveness performance.

3.4 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 regressors 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 it's 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 recovering them as repetition in the stream makes them more likely to be selected. Only following the oracle's first best pass would not help us learn to recover from errors.

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 correpsond 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 $\mathcal{A} = \{0, 1\}$ contains two actions with 0 indicating "skip the current sentence" and 1 indicating "extract the current sentence." A deterministic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. maps state-action pairs (s_t, a_t) to the next state s_{t+1}, i.e. updating the observed sentences with the next, i.e. $(t+1)$-th, sentence from the stream and updating the prefix of extraction variables with a_t. 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 repeatly apply $\hat{\pi}$ and take it's predicted action, stopping at state s_t. In the second phase we complete the trajectory using either the oracle π^* or $\hat{\pi}$ (the probability β of using π^* over $\hat{\pi}$ is a hyperparameter). Let π^{out} be the roll-out policy, 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}.

We also associate with $\mathbf{y}^{(s_t,a)}$ a non-negative cost $c(s_t, a)$ which connects 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

```
procedure LolsTraining
    Randomly initialize $\mathbf{W}$.
    repeat
        for $q \in \mathcal{Q}$ do
            $\mathcal{D} \leftarrow \varnothing$
            for $t = 1, \ldots, T$ do
                Roll-in by executing $\hat{\pi}$ for $t$ steps to arrive at state $s_t$.
                Set roll-out policy $\pi^{out} = \pi^*$ with probability $\beta$, otherwise $\hat{\pi}$.
                for $a \in \mathcal{A}$ do
                    Use $\pi^{out}$ to complete trajectory to obtain cost $c(s_t, a)$
                    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\big(\phi(s_t), c(s_t, a), a\big)\}$
                end for
            end for
            $\mathbf{W} = \mathbf{W} - \gamma \nabla \mathcal{L}(\mathbf{W})$
        end for
    until convergence
end procedure
```

Figure 5: The locally optimal learning to search (LOLS) training algorithm. γ is a learning rate.

costs, i.e. we minimize the function

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

using stochastic gradient descent.

The full LOLS training algorithm is presented in Figure 5. The benefit of using $\hat{\pi}$ to obtain the roll-in trajectory is that the oberved 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 can also learn the behavior of the more perfect oracle. The theoretical motivations of LOLS can be found in greater depth in Chang et al. (2015).

3.4.1 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. However, it will not always achieve the maximum possible $\mathbb{E}\left[\text{Gain}\right]$. For example, consider the sequence of sentences s_1, s_2, s_3, where nugget $n_1 \in s_1$, $n_2 \in s_2$, and $n_1, n_2 \in s_3$. The greedy oracle, proceeding sequentially would select sentences s_1 and s_2, and skip s_3, achieving an $\mathbb{E}\left[\text{Gain}\right]$ of $\frac{|\{n_1,n_2\}|}{|\{s_1,s_2\}|} = \frac{2}{2} = 1$. The maximum achievable $\mathbb{E}\left[\text{Gain}\right]$ is obtained by skipping the first two sentences and selecting sentence s_3 yielding $\frac{|\{n_1,n_2\}|}{|\{s_3\}|} = \frac{2}{1} = 2$. In practice we are far from matching the greedy oracle and so it suffices for now as an aspirational target.

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).

3.4.2 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 3.2 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. For example, it is not clear how to compare the document level PageRank of sentences from different documents. 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 3.2 with the nugget probability and document frequency separately, as well as together.

3.4.3 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 3.3 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 (see section 3.2) 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 1600%.

3.4.4 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 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 F-measure of $\mathbb{E}\left[\text{GAIN}\right]$ 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.

Cosine Similarity Threshold 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. 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(Team WATERLOOCLARKE at TREC 2015).

Salience-biased Affinity Propagation We also compare this model to a simplified varient 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.

Learning2Search+Cosine Similarity Threshold In this model, 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.

	unpenalized			latency-penalized			
	exp. gain	comp.	F_1	exp. gain	comp.	F_1	num. updates
Sap	**0.119**^{c}	0.09	0.094	0.105	0.088	0.088	8.333
Cos	0.075	0.176^{s}	0.099	0.095	0.236^{s}	0.128^{s}	145.615^{s,f}
L2s	0.097	**0.207**^{s,f}	0.112	0.136^{c}	**0.306**^{s,c,f}	0.162^{s}	89.872^{s,f}
L2sCos	0.115^{c,l}	0.189^{s}	**0.127**^{s,c,l}	**0.162**^{s,c,l}	0.276^{s}	**0.184**^{s,c,l}	29.231^{s,c}

Figure 6: 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).

	latency-penalized		
	exp. gain	comp.	F_1
Cos	0.095	**0.236**	0.128
LS-FS	0.164	0.220	0.157
LS-Cos-FS	**0.207**	0.18	**0.163**

Figure 7: Average system performance. LS-FS and LS-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.

3.4.5 Results

Results for system runs are shown in Figure 6. On average, L2s and L2sCos achieve higher F1 scores than the baseline systems in both latency penalized and unpenalized evaluations. For L2sCos, the difference in mean F1 score was significant compared to all other systems (for both latency settings).

Sap achieved the overall highest $\mathbb{E}[\textsc{Gain}]$, 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 6); 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 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 7 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 model (beyond similarity) of what to select is helpful. Ultimately we do much better when we can look at the whole document.

We also performed an error analysis to further understand how each system operates. Figure 8 shows the errors made by each system on the test streams. Errors were broken down into four categories. *Miss lead* and *miss body* errors occur when a system skips

	Miss	Miss			
	Lead	Body	Empty	Dupl.	Total
Sap	29.6%	68.7%	1.6%	0.1%	15,986
Cos	17.8%	39.4%	41.1%	1.7%	12,873
LS-FS	25.4%	71.7%	2.0%	0.9%	13,088
LS-Cos-FS	27.9%	70.8%	1.0%	0.2%	15,756
L2s	19.6%	55.3%	19.9%	5.1%	13,380
LS-Cos	24.6%	66.7%	7.5%	1.2%	11,613

Figure 8: Percent of errors made and total errors on test set.

a sentence containing a novel nugget in the lead or article body respectively. An *empty* error indicates an update was selected that contained no nugget. *Duplicate* errors occur when an update contains nuggets but none are novel.

Overall, errors of the miss type are most common and suggest future development effort should focus on summary content identification. About a fifth to a third of all system error comes from missing content in the lead sentence alone.

After misses, empty errors (false positives) are the next largest source of error. Cos was especially prone to empty errors (41% of its total errors). L2s is also vulnerable to empties (19.9%) but after applying the similarity filter and restriting to first sentences, these errors can be reduced dramatically (to 1%).

Surprisingly, duplicate errors are a minor issue in our evaluation. This is not to suggest we should ignore this component, however, as efforts to increase recall (reduce miss errors) are likely to require more robust redundancy detection.

4 Deep Learning Models of Content Salience

Deep learning methods have become the defacto 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 achitectures (Cheng and Lapata, 2016; Nallapati et al., 2016b,a; Narayan et al., 2018) thanks in part to the availibilty of a large corpus (approximately 300k) of CNN and Daily Mail articles with human written bullet point summaries (Hermann et al., 2015). Comparing models and defining best practices for model design has become difficult as papers often propose complex models with a variety of design choices, making it difficult to determine what choices actually lead to the best performance.

In this section, we describe completed experiments teasing out the importance of different neural network designs for sentence level salience (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 on our deep learning based models and find several impedements 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/topical content.

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 more useful for the summarization task than word window based embeddings.

We concluded this section with proposed domain adaptation experiments for modifying the word importance model to work on a news MDS task. The MDS version of the model will have an importance score aggregation step where word level scores accrue additional importance across documents using an attention mechanism.

4.1 Deep Learning Models of Sentence Salience

Given the diversity of neural architectural choices, a best practices for sentence extracive 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 definining 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 $\sum_{s\in\mathcal{E}} |s| \le c$.

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, via a mapping $W : \mathcal{V} \rightarrow \mathbb{R}^n$. The sentence encoder network $\textsc{Encoder} : \{\mathbb{R}^n\}^* \rightarrow \mathbb{R}^m$ is then responsible for mapping word embedding sequences to fixed length sentence embeddings. Finally, the sentence extractor network $\textsc{Extractor} : \{\mathbb{R}^m\}^* \rightarrow \{0,1\}^*$ 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). In the next sections, we describe the different encoder and extractor architectures, before discussing data, experiments, and evaluation.

4.1.1 Sentence Encoders

Averaging This encoder simply averages a sentence's associated word embeddings:

$$\textsc{Encoder}_{avg}(s) = \frac{1}{|s|}\sum_{w\in s} W(w).$$

Other than the word embeddings, this encoder involves no learned parameters, and while it collapses word order, embedding averaging has consistently been found competitive with more sophisticated sentence embedding techniques (Iyyer et al., 2015; Wieting et al., 2015; Arora et al., 2016; Wieting and Gimpel, 2017).

Recurrent Neural Networks The second encoder architecture we experiment with is a recurrent neural network (RNN) over the word embeddings. An RNN maintains an internal "hidden" state that is sequentially updated upon observing each word embedding. In practice, we use a bidirectional RNN with a gated recurrent unit (GRU) as the particular instantiation of the RNN cell (Cho et al., 2014). Under the RNN encoder, a sentence embedding for a sentence s is defined as the concatenation of the final forward and backward GRU outputs,

$$\textsc{Encoder}_{rnn}(s) = \left[\overrightarrow{\mathbf{h}}_{|s|}, \overleftarrow{\mathbf{h}}_{1}\right] \tag{1}$$

$$\overrightarrow{\mathbf{h}}_0 = \mathbf{0}; \quad \overrightarrow{\mathbf{h}}_i = \overrightarrow{\textsc{Gru}}\left(W(w_i), \overrightarrow{\mathbf{h}}_{i-1}\right) \tag{2}$$

$$\overleftarrow{\mathbf{h}}_{|s|+1} = \mathbf{0}; \quad \overleftarrow{\mathbf{h}}_i = \overleftarrow{\textsc{Gru}}\left(W(w_i), \overleftarrow{\mathbf{h}}_{i+1}\right) \tag{3}$$

where $[\cdot]$ is the concatenation operator; $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ are hidden states of the forward and backward GRU cells respectively; and $\overrightarrow{\textsc{Gru}}, \overleftarrow{\textsc{Gru}} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ are the forward and backward GRU cell operations. Nallapati et al. (2016a) use a bidirectional RNN for their sentence encoder.

Convolutional Neural Networks Our final sentence encoder uses a convolutional neural network (CNN) to encode salient n-gram windows into a fixed length vector. CNN's have grown increasingly popular in many NLP tasks as a computationally efficient substitute for RNN-based architectures (Kim, 2014; Lei et al., 2015; Dauphin et al., 2017). Our architecture largely follows Kim (2014): we apply a series of one dimensional convoluions over a sentence's word embeddings using varying width convolutions. For each convolutional window size $k \in K \subset \mathbb{N}$, a convolutional filter creates a feature vector $\mathbf{f}^{(k)} \in \mathbb{R}^{F_k}$ and the encoder output is the concatenation of the $|K|$ vectors. The set of filter window sizes K and the number of feature maps F_k for each $k \in K$ are model hyperparameters. Formally, we define the CNN encoding of a sentence s as

$$\textsc{Encoder}_{cnn}(s) = \left[\mathbf{f}^{(k)} : k \in K\right] \tag{4}$$

$$\mathbf{f}^{(k)}_l = \max_{i \in 1,\ldots,|s|-k+1} \textsc{ReLU}\left(a_i^{(l,k)}\right) \tag{5}$$

$$a_i^{(l,k)} = \mathbf{u}_l^{(k)} + \sum_{j=1}^{k} \mathbf{U}_{l,j}^{(k)} \cdot W(w_{i+j-1}) \tag{6}$$

where $\mathbf{u}^{(k)} \in \mathbb{R}^{F_k}$ and $\mathbf{U}^{(k)} \in \mathbb{R}^{k \times F_k \times n}$ are learned convolutional filter weights and $\textsc{ReLU}(x) = \max(0, x)$ is the rectified linear unit (Nair and Hinton, 2010). Cheng and Lapata (2016) use a CNN for their sentence encoder.

4.1.2 Sentence Extractors

A sentence extractor takes the encoder output, i.e. a sequence of sentence embeddings $\textsc{Encoder}(d) = \textsc{Encoder}(s_1), \ldots, \textsc{Encoder}(s_{|d|}) = \mathbf{h}_1, \ldots, \mathbf{h}_{|d|}$, and produces an extract label sequence $\mathbf{y}$. The sentence extractor is essentially a discriminative classifier

$p(\mathbf{y}_{1:|d|}|\mathbf{h}_{1:|d|})$. Previous neural network approaches 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 that we refer to as the RNN and Seq2Seq extractors.

SummaRunner Extractor Nallapati et al. (2016a) proposed a sentence extractor, which we refer to as the SummaRunner Extractor, that factorizes the extraction probability into contributions from different sources. First, a bidirectional RNN is run over the sentence embeddings[2] and the output is concatenated. A representation of the whole document is made by averaging the RNN output. A summary representation is also constructed by taking the sum of the previous RNN outputs weighted by their extraction probabilities. Extraction predictions are made using the RNN output at the i-th prediction step, the document representation, and i-th version of the summary representation, along with factors for sentence location in the document. The use of the iteratively constructed summary representation creates a dependence of $\mathbf{y}_i$ on all $\mathbf{y}_{<i}$.

In more detail, first the bidirectional RNN is applied to the input sentence embeddings,

$$\overrightarrow{\mathbf{z}}_0 = \mathbf{0} \qquad \overrightarrow{\mathbf{z}}_i = \overrightarrow{\textsc{Gru}}(\mathbf{h}_i, \overrightarrow{\mathbf{z}}_{i-1}) \tag{7}$$

$$\overleftarrow{\mathbf{z}}_{|d|+1} = \mathbf{0} \qquad \overleftarrow{\mathbf{z}}_i = \overleftarrow{\textsc{Gru}}(\mathbf{h}_i, \overleftarrow{\mathbf{z}}_{i+1}). \tag{8}$$

Then a document embedding $\mathbf{d}$ is created by averaging in the hidden RNN states and passing them through a feedforward layer:

$$\mathbf{d} = \tanh\left(\mathbf{b}_d + \mathbf{W}_d \frac{1}{|d|} \sum_{i=1}^{|d|} \left[\overrightarrow{\mathbf{z}}_i; \overleftarrow{\mathbf{z}}_i\right]\right). \tag{9}$$

The RNN outputs are then passed through a separate fully connected layer to create a sentece representation $\mathbf{z}_i$ where

$$\mathbf{z}_i = \textsc{ReLU}\left(\mathbf{b}_z + \mathbf{W}_z \left[\overrightarrow{\mathbf{z}}, \overleftarrow{\mathbf{z}}\right]\right) \tag{10}$$

The extraction probability is then determined by contributions from five sources:

$$\textit{sentence salience} \quad a_i^{(sent)} = \mathbf{W}^{(sent)}\mathbf{z}_i, \tag{11}$$

$$\textit{document salience} \quad a_i^{(doc)} = \mathbf{z}_i^T \mathbf{W}^{(doc)}\mathbf{d}, \tag{12}$$

$$\textit{summary novelty} \quad a_i^{(nov)} = -\mathbf{z}_i^T \mathbf{W}^{(nov)}\mathbf{s}_i, \tag{13}$$

$$\textit{fine-grained position} \quad a_i^{(fpos)} = \mathbf{W}^{(fpos)}\mathbf{f}_i, \tag{14}$$

$$\textit{coarse-grained position} \quad a_i^{(cpos)} = \mathbf{W}^{(cpos)}\mathbf{c}_i, \tag{15}$$

where $\mathbf{f}_i$ and $\mathbf{c}_i$ are embeddings associated with the i-th sentence position and the quarter of the document containing sentence i respectively. In Equation 13, $\mathbf{s}_i$ is a representation

[2] Nallapati et al. (2016a) use an RNN sentence encoder with this extractor architecture; in this work we pair the SummaRunner extractor with different encoders.

of the summary after making the first $i-1$ predictions, and is computed as the sum of the previous $\mathbf{z}_{<i}$ weighted by their extraction probabilities,

$$\mathbf{s}_i = \tanh\left(\sum_{j=1}^{i-1} p(\mathbf{y}_j = 1|\mathbf{y}_{<j}, \mathbf{h}_{1:|d|}) \cdot \mathbf{z}_j\right), \qquad \mathbf{s}_1 = \mathbf{0}. \tag{16}$$

Note that the presence of this term induces dependence of each $\mathbf{y}_i$ to all $\mathbf{y}_{<i}$. The final extraction probability is the logistic sigmoid of the sum of these terms plus a bias,

$$p(\mathbf{y}_i = 1|\mathbf{y}_{<i}, \mathbf{h}_{1:|d|}) = \sigma\left(a_i^{(sent)} + a_i^{(doc)} + a_i^{(nov)} + a_i^{(fpos)} + a_i^{(cpos)} + b\right). \tag{17}$$

The weight matrices $\mathbf{W}_d$, $\mathbf{W}_z$, $\mathbf{W}^{(sent)}$, $\mathbf{W}^{(sal)}$, $\mathbf{W}^{(nov)}$, $\mathbf{W}^{(fpos)}$, $\mathbf{W}^{(cpos)}$ and bias terms $\mathbf{b}_d$, $\mathbf{b}_z$, and b are learned parameters; additionally, the GRUs have separate learned parameters.

RNN Extractor (Kedzie et al., 2018) Our first proposed model is a very simple bidirectional RNN based tagging model, and can be thought of as a bare bones version of the SummaRunner extractor. In this model, we pass the outputs of a bidirectional GRU over the sentence embeddings into a multi-layer perceptron that terminates in a layer of sigmoid activations corresponding to the sentence level extraction probabilities.

$$\overrightarrow{\mathbf{z}}_0 = \mathbf{0} \qquad \overrightarrow{\mathbf{z}}_i = \overrightarrow{\textsc{Gru}}(\mathbf{h}_i, \overrightarrow{\mathbf{z}}_{i-1}) \tag{18}$$

$$\overleftarrow{\mathbf{z}}_{|d|+1} = \mathbf{0} \qquad \overleftarrow{\mathbf{z}}_i = \overleftarrow{\textsc{Gru}}(\mathbf{h}_i, \overleftarrow{\mathbf{z}}_{i+1}) \tag{19}$$

$$\mathbf{o}_i = \textsc{ReLU}\left(\mathbf{U} \cdot [\overrightarrow{\mathbf{z}}_i; \overleftarrow{\mathbf{z}}_i] + \mathbf{u}\right) \tag{20}$$

$$p(\mathbf{y}_i = 1|\mathbf{h}_{1:|d|}) = \sigma\left(\mathbf{v} \cdot \mathbf{o}_i + v\right) \tag{21}$$

where $\overrightarrow{\textsc{Gru}}$ and $\overleftarrow{\textsc{Gru}}$ indicate the forward and backward GRUs respectively, and each have separate learned parameters; $\mathbf{U}, \mathbf{v}$ and u, v are learned weight and bias parameters respectively.

Cheng & Lapata Extractor This extractor, proposed by Cheng and Lapata (2016), is built around a sequence-to-sequence model. First, each sentence embedding[3] is fed into an encoder side RNN, with the final encoder state passed to the first step of the decoder RNN. On the decoder side, the same sentence embeddings are fed as input to the decoder and both encoder and decoder outputs are used to predict each $\mathbf{y}_i$. The decoder input is weighted by the previous extraction probability, inducing the dependence of $\mathbf{y}_i$ on $\mathbf{y}_{<i}$.

The basic architecture is a sequence-to-sequence model defined as follows:

$$\mathbf{e}_0 = \mathbf{0} \qquad \mathbf{e}_i = \textsc{Gru}_{enc}(\mathbf{h}_i, \mathbf{e}_{i-1}) \tag{22}$$

$$\mathbf{d}_1 = \textsc{Gru}_{dec}(\mathbf{h}_*, \mathbf{e}_{|d|}) \qquad \mathbf{d}_i = \textsc{Gru}_{dec}(p_{i-1} \cdot \mathbf{h}_{i-1}, \mathbf{d}_{i-1}) \tag{23}$$

$$\mathbf{o}_i = \textsc{ReLU}\left(\mathbf{U} \cdot [\mathbf{e}_i; \mathbf{d}_i] + \mathbf{u}\right) \tag{24}$$

$$p_i = p(\mathbf{y}_i = 1|\mathbf{y}_{<i}, \mathbf{h}_{1:|d|}) = \sigma\left(\mathbf{v} \cdot \mathbf{o}_i + v\right) \tag{25}$$

[3] Cheng and Lapata (2016) used an CNN sentence encoder with this extractor architecture; in this work we pair the Cheng & Lapata extractor with several different encoders.

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

where $\mathbf{h}_*$ is a learned "begin decoding" sentence embedding; each GRU has separate learned parameters; and $\mathbf{U}, \mathbf{v}$ and u, v are learned weight and bias parameters. Note in Equation 27 that the decoder side GRU input is the sentence embedding from the previous time step weighted by its probabilitiy of extraction (p_{i-1}) from the previous step, inducing dependence of each output $\mathbf{y}_i$ on all previous outputs $\mathbf{y}_{<i}$.

Seq2Seq Extractor (Kedzie et al., 2018) Our second proposed model, the represents a more typical sequence-to-sequence architecture with dot product style attention (Luong et al., 2015), used for neural machine translation Bahdanau et al. (2014) and abstractive summarization See et al. (2017). The sentence embeddings are first encoded by a bidirectional Gru. A separate decoder Gru transforms each sentence into a query vector which attends to the encoder output. The attention weighted encoder output and the decoder Gru output are concatenated and fed into a multi-layer perceptron to compute the extraction probability. Unlike the Cheng & Lapata extractor, the Seq2Seq extractor does not induce dependencies between the extraction labels, i.e. they are conditionally independent.

The architecture is defined as follows

$$\mathbf{e}_0 = \mathbf{0} \qquad \mathbf{e}_i = \textsc{Gru}_{enc}(\mathbf{h}_i, \mathbf{e}_{i-1}) \tag{26}$$

$$\mathbf{d}_0 = \textsc{Gru}_{dec}(\mathbf{h}_*, \mathbf{e}_{|d|}) \qquad \mathbf{d}_i = \textsc{Gru}_{dec}(\mathbf{h}_i, \mathbf{d}_{i-1}) \tag{27}$$

$$\alpha_{i,j} = \frac{\exp\left(\mathbf{d}_i \cdot \mathbf{e}_j\right)}{\sum_{j=1}^{|d|} \exp\left(\mathbf{d}_i \cdot \mathbf{e}_j\right)} \qquad \bar{\mathbf{e}}_i = \sum_{j=1}^{|d|} \alpha_{i,j} \mathbf{e}_j \tag{28}$$

$$\mathbf{o}_i = \textsc{ReLU}\left(\mathbf{U} \cdot [\bar{\mathbf{e}}_i; \mathbf{d}_i] + \mathbf{u}\right) \tag{29}$$

$$p(\mathbf{y}_i = 1 | \mathbf{h}_{1:|d|}) = \sigma\left(\mathbf{v} \cdot \mathbf{o}_i + v\right) \tag{30}$$

where $\mathbf{h}_*$ is a learned "begin decoding" sentence embedding; each GRU has separate learned parameters; and $\mathbf{U}, \mathbf{v}$ and u, v are learned weight and bias parameters. In practice, we run this model in "bidirectional mode" where seperate sequence-to-sequence model also run in the reverse direction, equations 28-29 are computed using the concatenation of the encoder/decoder outputs.

4.1.3 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 **??** for dataset statistics.

CNN-DailyMail We use the preprocessing and training, validation, and test splits of See et al. (2017). This corpus is a mix of news on different topics including politics, sports, and entertainment.

New York Times The New York Times (NYT) corpus (Sandhaus, 2008) contains two types of abstracts for a subset of its articles. The first summary is an archival abstract and the second is a shorter online teaser meant to entice a viewer of the webpage to click to read more. From this collection, we take all articles that have a concatenated summary length of at least 100 words. We create training, validation, and test splits by partitioning on dates; we use the year 2005 as the validation data, with training and test partitions including documents before and after 2005 respectively.

DUC We use the single document summarization data from the 2001 and 2002 Document Understanding Conferences (DUC) (Over and Liggett, 2002). We split the 2001 data into training and validation splits and reserve the 2002 data for testing.

AMI The AMI corpus (Carletta et al., 2005) is a collection of real and staged office meetings annotated with text transcriptions, along with abstractive summaries. We use the prescribed splits.

Reddit Ouyang et al. (2017) collected a corpus of personal stories shared on Reddit[4] along with multiple extractive and abstractive summaries. We randomly split this data using roughly three and five percent of the data validation and test respectively.

PubMed We created a corpus of 25,000 randomly sampled medical journal articles from the PubMed Open Access Subset[5]. We only included articles if they were at least 1000 words long and had an abstract of at least 50 words in length. We used the article abstracts as the ground truth human summaries.

Ground Truth Extract Summaries Since we do not typically have ground truth extract summaries from which to create the labels $\mathbf{y}_i$, we construct gold label sequences by greedily optimizing Rouge-1 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 ?Durrett et al. (2016); Nallapati et al. (2016a) which found this to be the easier target to learn.

4.1.4 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 question by evaluating Rouge recall and Meteor (Denkowski and Lavie, 2014) performance across

[4] `www.reddit.com`

[5] `https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/`

our six collected datasets. We perform the standard stochastic gradient descent based optimization (using the Adam update (Kingma and Ba, 2014)) of the weighted negative log likehood

$$\mathcal{L}(\theta) = -\sum_{d,\mathbf{y}\in\mathcal{D}} \sum_{i} \omega(\mathbf{y}_i) \log p(\mathbf{y}_i|\mathbf{y}_{<i}, \textsc{Encoder}(d); \theta)$$

where θ are model parameters and $\omega(\mathbf{y}_i)$ upweights the positive labels to account for the imbalanced label distribution.

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 with out 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 news domain (Nenkova, 2005).

4.1.5 Results

Ext.	Enc.	**CNN/DM**		**NYT**		**DUC 2002**		**Reddit**		**AMI**		**PubMed**	
		M	R-2	M	R-2	M	R-2	M	R-2	M	R-2	M	R-2
Lead	–	24.1	24.4	30.0	32.3	25.1	21.5	**20.1**	**10.9**	12.3	2.0	15.9	9.3
RNN	Avg.	**25.2**	25.4	29.8	34.7	**26.8**	22.7	**20.4**	**11.4**	**17.0**	**5.5**	19.8	17.
	RNN	25.1	25.4	29.6	34.9	**26.8**	22.6	**20.2**	**11.4**	16.2	**5.2**	19.7	16.
	CNN	25.0	25.1	29.0	33.7	**26.7**	**22.7**	**20.9**	**12.8**	14.4	3.2	19.9	16.
Seq2Seq	Avg.	**25.2**	**25.6**	**30.5**	**35.7**	**27.0**	**22.8**	**20.9**	**13.6**	**17.0**	**5.5**	**20.1**	**17.**
	RNN	**25.1**	25.3	30.2	**35.9**	**26.7**	22.5	**20.5**	**12.0**	16.1	**5.3**	19.7	16.
	CNN	25.0	25.1	29.9	35.1	**26.7**	**22.7**	**20.7**	**13.2**	14.2	2.9	19.8	16.
Cheng & Lapata	Avg.	25.0	25.3	30.4	**35.6**	**27.1**	**23.1**	**20.9**	**13.6**	**16.7**	**6.1**	**20.1**	**17.**
	RNN	25.0	25.0	**30.3**	**35.8**	**27.0**	**23.0**	**20.3**	**12.6**	**16.3**	**5.0**	19.7	16.
	CNN	**25.2**	25.1	29.9	35.0	**26.9**	**23.0**	**20.5**	**13.4**	14.3	2.8	19.9	16.
Summa Runner	Avg.	25.1	25.4	30.2	35.4	26.7	22.3	**21.0**	**13.4**	**17.0**	**5.6**	19.9	17.
	RNN	25.1	25.2	30.0	35.5	26.5	22.1	**20.9**	**12.5**	**16.5**	**5.4**	19.7	16.
	CNN	24.9	25.0	29.3	34.4	26.4	22.2	**20.4**	**12.3**	14.5	3.2	19.8	16.
Oracle	–	31.1	36.2	35.3	48.9	31.3	31.8	24.3	16.2	8.1	3.9	24.1	25.

Table 5: METEOR (M) and ROUGE-2 recall (R-2) results across all extractor/encoder pairs. Results that are statistically indistinguishable from the best system are shown in bold face.

Ext.	Emb.	CNN/DM		NYT		DUC		Reddit		AMI		PubMed	
Seq2Seq	Fixed	**25.6**		**35.7**		**22.8**		**13.6**		5.5		**17.7**	
	Learn	25.3	(0.3)	**35.7**	(0.0)	**22.9**	(-0.1)	**13.8**	(-0.2)	**5.8**	(-0.3)	16.9	(0.8)
C&L	Fixed	**25.3**		**35.6**		**23.1**		**13.6**		**6.1**		**17.7**	
	Learn	24.9	(0.4)	35.4	(0.2)	**23.0**	(0.1)	**13.4**	(0.2)	**6.2**	(-0.1)	16.4	(1.3)
Summa Runner	Fixed	**25.4**		**35.4**		**22.3**		**13.4**		**5.6**		**17.2**	
	Learn	25.1	(0.3)	35.2	(0.2)	**22.2**	(0.1)	12.6	(0.8)	**5.8**	(-0.2)	16.8	(0.4)

Table 6: 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. All extractors use the averaging sentence encoder. When both learned and fixed settings are bolded, there is no signifcant performance difference. RNN extractor is omitted for space but is similar to Seq2Seq. Difference in scores shown in parenthesis.

The results of our main experiment comparing the different extractors/encoders are shown in Table 5. Overall, we find no major advantage when using the CNN and RNN sentence encoders over the averaging encoder. 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.

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 configuration that is indistinguishable from the best system on the AMI corpus. This is probably due to the small test size of these datasets.

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 6 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.[6] This suggests that we cannot extract much generalizable learning

[6] The AMI corpus is an exception here where learning *does* lead to small performance boosts, however,

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† (0.1)	34.3† (0.4)	22.3† (0.4)	10.3† (1.1)	3.8† (1.7)	15.7† (1.3)
-verbs	25.3† (0.1)	34.4† (0.3)	22.4† (0.3)	10.8 (0.6)	5.8 (-0.3)	16.6† (0.4)
-adj/adv	25.3† (0.1)	34.4† (0.3)	22.5 (0.2)	9.5† (1.9)	5.4 (0.1)	16.8† (0.2)
-function	25.2† (0.2)	34.5† (0.2)	**22.9**† (-0.2)	10.3† (1.1)	**6.3**† (-0.8)	16.6† (0.4)

Table 7: 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. Difference in score from *all words* shown in parenthesis.

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). All datasets were automatically tagged using the spaCy part-of-speech (POS) tagger[7]. 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 7 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.

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[8]; 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

only in the Seq2Seq extractor is this diference significant; it is quite possible that this is an artifact of the very small test set size.

[7] https://github.com/explosion/spaCy

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

Ext.	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	(3.9)	25.6	(10.1	21.2	(1.6)	**13.5**	(0.1)	**6.0**	(-0.5)	14.9	(2.8)

Table 8: 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.

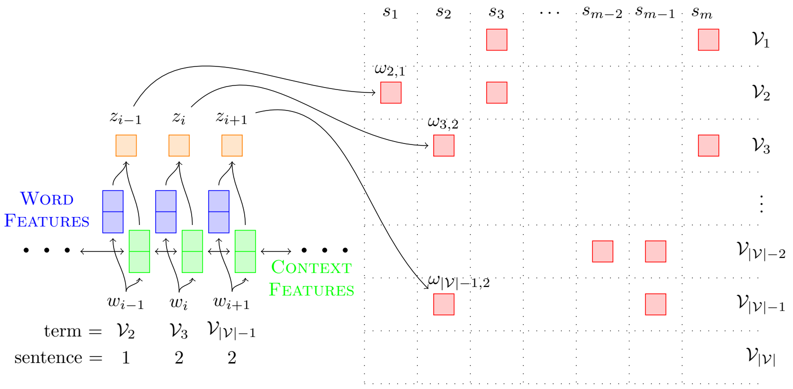

Figure 9: Left: Word and context features are extracted from the flat token sequence representation to get word level importance scores z_i. Right: word level scores are aggregated into a sparse bag-of-words representation of the input sentences.

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 8 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 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.

4.2 Word Importance Estimation in Deep Learning Models

Our previous experiments revealed that lexical sematics were not the main driver of learning in sentence extractive news summarization. One could plausibly argue that it is a feature, not a bug, and that the structural signals in news are intentional and not to be avoided. However, we think more attention could be paid to estimating importance scores at the word level. We are motivated by potential application to abstractive generation:

better word level importance estimation could help to remove all but the most necessary content from the documents as a preprocessing stage before abstractive summarization. We are also encouraged by practices in multi-document news summarization, where word importance weights are the main ingredient in sentence representations.

Occams and its antecedent Classy have been consistent top performers in various summarization workshops (Conroy et al., b,a; Davis et al., 2012). In general their main approach is to represent each sentence as a sparse bag-of-words, where non-zero entries correspond to word importance weights for the words found in the sentence. Typically, tf-idf weights are used for the importance scores. The term by sentence matrix representing the document or documents to be summarized is then factorized into two matrices (typically with non-negative entries) representing term factors and sentence factors. The entries in the sentence factor matrix represent latent "topics," and apriori the importance of each sentence is the sum of its latent topic entries.

Sentence selection can subsequently be performed using one of several methods. In the naive case, one can select the sentence with the highest vector norm, substract the selected latent factors from the remaining sentence vectors (zeroing out any terms that become negative), and repeating until the summary length budget is reached. More sophisticated selection procedures involving multi-dimensional knapsack packing or submodular optimization can be used, however these are not the focus of this work.

One draw back to this approach is that the sentence factors and word importance scores are unsupervised with respect to the final summarization objective; their utility to the summarization task is a happy coincidence. Additionally, the word importance scores are not assigned based on the context in which the word appears.

We propose to address these issues by learning word level importance scores in the process of single document sentence extractive summarization. Additionally, we propose a method of adapting these scores from the single document case to multi-document summarization.

4.2.1 Proposed Model

Let $\mathcal{V}$ be a finite vocabulary of words. A document d is a sequence of m words $(w_1, w_2, \ldots, w_m) \in \mathcal{V}^m$. We define two mappings of words to dense vector representations. The first $\textsc{Features} : \mathcal{V} \rightarrow \mathbb{R}^{n_f}$ 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, and other word features that have been shown to be useful for summarization. The second mapping $\textsc{Context} : \mathcal{V} \rightarrow \mathbb{R}^{n_c}$ maps the word to it's 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 feedforward layer

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

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 ω_i be the BOW

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

Figure 10: Simple sentence extraction algorithm given the sentence BOW representations ω_i, a word budget β, and a vector κ of sentence lengths (in words) as input.

representation of the i-th sentence with entries

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

for all $j \in \{1, \ldots, |\mathcal{V}|\}$.

With the BOW representations in hand, we perform sentence selection using the algorithm presented in Figure 10 to obtain a 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

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

with respect to the parameters θ of the word importance predictor. If needed, we 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 and any human reference abstract and 0 otherwise. The $\mathcal{L}_{ext}$ would then be augmented with an additional cross entropy loss for the word level predictions:

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

Adaptation to MDS We also propose a simple self attention-based modification to the word importance aggregation step to help adapt this method to multi-document summarization (MDS). 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).

We plan to experiment with the following importance aggregation method. First, given the outputs of the contextual features $\mathbf{h}_i = \textsc{Context}(w_i)$, 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 $\bar{z}_i$ inplace of their nonaggregated counterparts in the creating the BOW representations ω_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 effectively not contribute much to a sentences score, reducing the effectve dimension of the BOW vectors and regularizing individual sentences to the document cluster's mean.

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.

5 Faithful Text Generation

Sections 3 and 4 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. Abstractive text generation adds significant challenges to summary creation, but 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.

As a potential solution, we propose modeling text generation as a two player game between the generator and recognizer, akin to an auto-encoder (Rumelhart et al., 1985). First, we provide as input to the generator some evidence (e.g., raw text or structured data like entries from a database). Conditioned on the evidence, the generator must produce a list of candidate utterances describing the evidence. The recognizer reranks the candidates based on the likelihood of predicting the correct evidence. While we will experiment with a variety of loss functions to find something that works well empirically, we expect in principal to train the generator to maximize the expected likelihood of evidence under the recognizer.

See Figure 11 for an example where we have biographical data (structured data about name, nationality, occupation, etc.), and we generate several plausible and implausible candidates that are evaluated by the recognizer. The first candidate is ranked highly

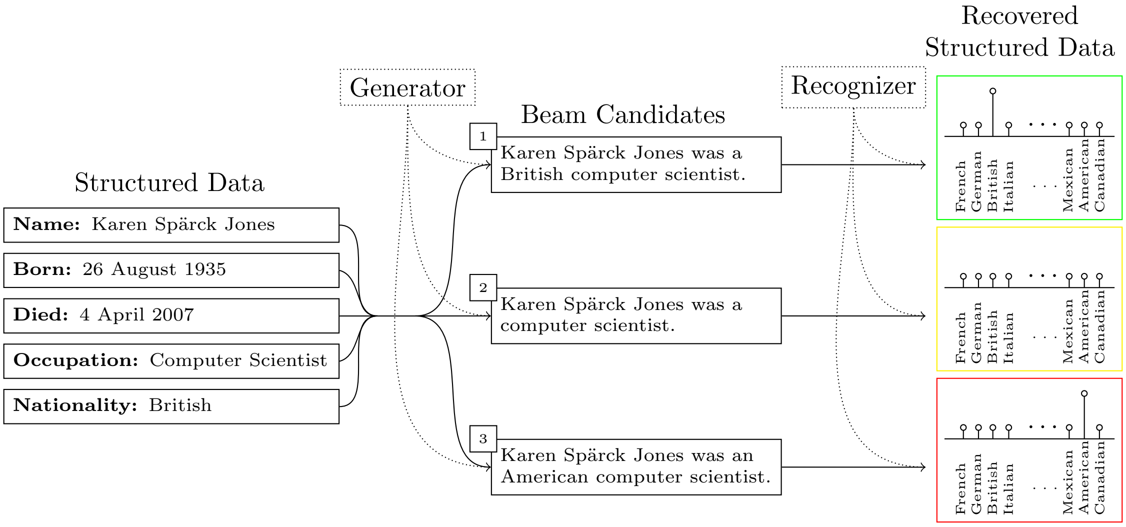
Structured Data
Name: Karen Spärck Jones
Born: 26 August 1935
Died: 4 April 2007
Occupation: Computer Scientist
Nationality: British
Generator
Beam Candidates
1
Karen Spärck Jones was a British computer scientist.
2
Karen Spärck Jones was a computer scientist.
3
Karen Spärck Jones was an American computer scientist.
Recognizer
Recovered Structured Data
French
German
British
Italian
...
Mexican
American
Canadian

Figure 11: 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.

because the recognizer can correctly infer that the nationality of Karen Spärck Jones is British (green box). The second candidate is possibly ok 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 structured data (red box).

In the remainder of this section, we formally define two faithful generation models, one for generating text from structured data and one for generating text from text data. We conclude by briefly describing some applications and datasets we will use to evaluate these modls.

5.1 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.

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 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 a faithful proxy for the underlying document, 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 repsonsible 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 logical 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.

5.2 Structured Data Model

Our structured data 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 $\mathcal{X}_i = \{\textsc{Accountant}, \textsc{Actor}, \ldots, \textsc{Zoologist}\}$. 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 faithul 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 faithul with the quantity

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

where $\beta(x) \propto \frac{p(y|x;\theta)}{|y|}$ is a the generator distribution renormalized by average token probability over the beam candidates $y^{(1)}, \ldots, y^{(k)}$.

We implement the recognizer using CNN classifiers with parameters ϕ, and fit the parameters with the average log likehood objective

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

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

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

where the second term results in a minimum risk training (MRT) style gradient update (Edunov et al., 2018).

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

Figure 12: Example Summarization Cloze Question

5.3 Text Data Model

For cases where the evidence is not structured data but text, e.g. summarization, we can modify the structured data model slightly to obtain a workable faithful training regime. Our data will 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 12 for an example from the CNN/DailyMail dataset.

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

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

where the entropy terms are defined similarly to subsection 5.2 and the recognizer is now modified to map cloze question/passage tuples $\bar{x}, y$ to probabilties 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 5.2.

5.4 Applications and Data

We envision several applications and scenarios where faithful generation might be useful. The first is for data-to-text generation tasks. We plan on training the structured data model on at least 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). For the text-to-text scenarios, 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 tractible 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).

We would also like to explore ways in which faithful generation could be applied to understanding and controlling for bias in machine learning (Bolukbasi et al., 2016). By modifying the faithful generation objective function to reward only uncertainty for missing data fields, we could encourage the generator to produce utterances that are decorrelated with a particular field value. As an example let's say that our observed evidence consists of x_{name} = Fran Allen and $x_{occupation}$ = Computer Scientist but we did not know the value of the gender field, x_{gender} = Missing. If our dataset over represents men (a known bias in Wikipedia), the sequence-to-sequence based generator is likely to fall back to the most prevalent patterns in the dataset and use the masculine pronoun during generation. In turn, the gender field recognizer is likely to key in on pronoun usage as a discriminative feature and predict male. If we train the faithful generator to produce utterances that result in maximum uncertainty in the gender recognizer then it is likely to avoid gendered pronouns and minimize the chances of implying incorrect knowledge.

6 Research Plan

Research plan.

7 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 spans 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. 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. We also believe that faithful generation can be useful on a variety of text generation tasks beyond just summarization, e.g. data-to-text generation.

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.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. 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.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357, 2016.

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`.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. The ami meeting corpus: A pre-announcement. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 28–39. Springer, 2005.

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.

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.

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

James M Conroy, Jjudith D Schlesinger, PO Dianne, Mary E Okurowski, et al. Using {HMM} and logistic regression to generate extract summaries for {DUC}. 2001.

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, Judith D Schlesinger, Jeff Kubina, Peter A Rankel, and Dianne P OLeary. Classy 2011 at tac: Guided and multi-lingual summaries and evaluation metrics. a.

John M Conroy, Judith D Schlesinger, Dianne P OLeary, and Jade Goldstein. Back to basics: Classy 2006. b.

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

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, pages 933–941, 2017.

Sashka T Davis, John M Conroy, and Judith D Schlesinger. Occams–an optimal combinatorial covering algorithm for multi-document summarization. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining Workshops*, pages 454–463. IEEE Computer Society, 2012.

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.

Delbert Dueck. *AFFINITY PROPAGATION: CLUSTERING DATA BY PASSING MESSAGES*. PhD thesis, University of Toronto, 2009.

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 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.

David Graff and C Cieri. English gigaword corpus. *Linguistic Data Consortium*, 2003.

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.

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.

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.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691, 2015.

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.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Molding cnns for text: non-linear, non-consecutive convolutions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1565–1575, 2015.

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.

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.

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.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

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.

Jessica Ouyang, Serina Chang, and Kathy McKeown. Crowd-sourced iterative annotation for narrative summarization corpora. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 46–51, 2017.

Paul Over and Walter Liggett. Introduction to duc: An intrinsic evaluation of generic news text summarization systems. *Proc. DUC. http://wwwnlpir. nist. gov/projects/duc/guidelines/2002. html*, 2002.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. 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.

Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization-Volume 4*, pages 21–30. Association for Computational Linguistics, 2000.

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.

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.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

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

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

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.

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.

Andreas Stolke. Srilm-an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 901–904, 2002.

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

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.

John Wieting and Kevin Gimpel. Revisiting recurrent networks for paraphrastic sentence embeddings. *arXiv preprint arXiv:1705.00364*, 2017.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 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, Stuart Shieber, and Alexander Rush. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, 2017.