Nutshell: A Biased LexRank Topic-focused Multi-document Extractive Summarization System

Shannon Ladymon Haley Lepp Benny Longwill Amina Venton
Department of Linguistics, University of Washington
{sladymon,hlepp,longwill,aventon}@uw.edu

Abstract

We design and implement Nutshell, an automatic topic-focused multi-document extractive summarization system. Nutshell cleans and compresses sentences using linguistically-informed rule-based patterns which extract and remove clauses and punctuation found in other literature to be less valuable in a summary. Nutshell then uses the Biased LexRank graph algorithm to calculate sentence salience via inter-sentential similarity and topic-sentence bias. The ranked sentences are then selected greedily based on this score while avoiding redundancy and remaining within the 100 word limit. Users can choose between chronological and entity-based sentence ordering. Nutshell outperforms the LEAD and MEAD baseline ROUGE-2 scores on both TAC 2010 and TAC 2011.

1 Introduction

In this study, we describe Nutshell, an end-to-end automatic multi-document summarization system that takes in topic-oriented document clusters from ACQUAINT, ACQUAINT2, and Gigaword, and produces 100-word extractive summaries for each topic cluster for the Text Analysis Conferences (TAC) shared task evaluations for 2010 and 2011. Nutshell makes use of linguistically-informed rules as described in (Zajic et al., 2007; Vanderwende et al., 2007; Wang et al., 2013) to compress sentences to their most potent semantic content, and then uses the Biased LexRank graph approach to determine sentence salience using inter-sentential similarity weighting with a bias for the cluster topic (Erkan and Radev, 2004; Otterbacher et al., 2005, 2009). Once ranked, sentences are selected greedily with constraints to eschew redundancy and ensure correct summary length.

Nutshell provides the option of chronological and entity-based ordering of extracted sentences. Chronological ordering organizes sentences by date of document and order within document. Entity-based ordering (Barzilay and Lapata, 2008) learns the distribution of entity mentions across sentences from human-generated gold summaries, and then builds a classification support vector machine (SVM) model to predict the best ordering of sentences for machine-generated summaries.

Because the components of Nutshell each offer the user multiple configuration options, we test different variations of the system to determine the values that will produce the optimal ROUGE-2 scores.

We evaluate the output summaries against human-produced summaries from the TAC 2010 and 2011 tasks using ROUGE-1 and ROUGE-2 metrics. We find that Nutshell outperforms the LEAD and MEAD baselines on TAC 2010 and TAC 2011 and has comparable readability to peer systems.

2 System Overview

Nutshell follows a four-component pipeline (Data Preprocessing, Content Realization, Content Selection, and Information Ordering) as shown in Figure 1. The system allows flexibility in choosing options and formulas for different modules.

First, using the input document clusters, Nutshell preprocesses the XML files. Nutshell then undergoes content realization and generates cleaned and compressed candidate sentences using a rule-based compression approach. Users have the option to choose from seven different linguistically-motivated cleaning and compression rules that include removing headers, parentheti-

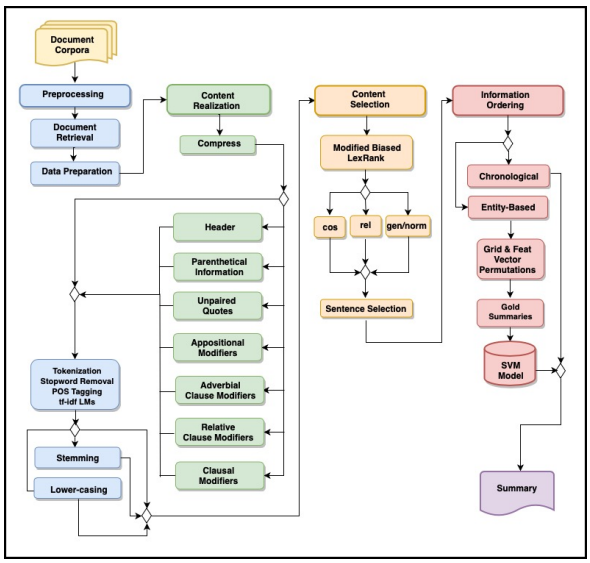

Figure 1: System Architecture

cal information, unpaired quotation marks, noun appositives, clausal modifiers of nouns, relative clause modifiers, and adverbial clausal modifiers. Sentences are cleaned and compressed using pattern-matching and a dependency parser to identify grammatical constituents. All of the candidate sentences, including the original, then undergo further preprocessing to parse the data for tokenization, stop-word removal, and POS tagging. We create a tf-idf language model dependent on user selection. The preprocessing module also has options for stemming and lower-casing.

To select content, Nutshell uses Biased LexRank to score candidate sentences by salience and relevance to the cluster as well as to the topic. There are options to calculate the Biased LexRank using formulas for cosine similarity, relevance, or (normalized) generative probability. Using the tf-idf model built in the preprocessing module and the similarity formula, Nutshell calculates Bias LexRank scores. Ranked by score, sentences are then filtered for redundancy and selected iteratively until the summary reaches a maximum length.

This summary is then passed through information ordering, which gives users the option to order the sentences chronologically or through an entity-based approach. Chronological ordering arranges sentences by date (most recent news articles last) and by position of the sentence in the original document. Entity-based ordering trains a model on human-created summaries to find the most optimal sentence ordering. The model is generated by an SVM using extracted linguistic features from sentences used to create entity grids and feature vectors. The sentence ordering with the highest model score ranking is selected to use for the summary. Finally, the summary is directly output and evaluated using ROUGE metrics.

3 Approach

Below, we describe our approach in four subcomponents: Data Preparation, Content Realization, Content Selection, and Information Ordering.

3.1 Data Preparation

For the TAC 2010 and 2011 tasks, we collect data from a set of NIST standard XML files that identify a set of training topics. First a Document Retriever is used to extract and parse the text, and then pre-processing occurs after the Content Realization module creates compressed versions of each sentence.

3.1.1 Document Retrieval

A Document Retriever object extracts raw document HTML or XML. We re-configure the source and publication date from the document IDs to derive a file path to the respective document corpora. Depending on the year of production, we assemble the file path for the ACQUAINT (Graff, 2002) or ACQUAINT2 (Vorhees and Graff, 2008) corpora as well as the English Gigaword corpus (Parker et al., 2011) and make necessary modifications to adhere to file path naming conventions. These documents are then parsed using the Beautiful Soup Python library (Nair, 2014).

3.1.2 Pre-Processing

After Content Realization (discussed in section 3.2) provides one or more versions of each original sentence, NLTK (Loper and Bird, 2002) is used to instantiate Sentence and Token Class Objects, and to remove stopwords using the NLTK.corpus stopwords list.

Two pre-processing options are integrated using Boolean flags. The first of these methods is stemming, a technique that utilizes the NLTK PorterStemmer to remove morphological affixes from words through suffix-stripping. Although the PorterStemmer is one of the oldest stemmers, having been developed in 1979 (Porter, 1980), we chose it due to its simplicity and speed. We also implemented an option to test lower casing input text separately.

Rule	Example
Header	**[LITTLETON, Colo. (AP) –]** Students returned to classes...
Parenthetical Information	It has applied to the International Whaling Commission **[(IWC)]**...
Unpaired quotes	**[“]**The coral reef system might be totally destroyed.
Noun Appositives	...said Tyler Herbert, **[16, a sophomore]**.
Adverbial Clause Modifiers	**[As they gave periodic updates through the night]**, Davis and Stone emphasized...
Relative Clause Modifiers	...a massacre by two students at Columbine High, **[whose teams are called the Rebels]**,...
Clausal Modifiers of Nouns	Some of the bombs **[used in the assault at Columbine]** were planted in knapsacks.

Note: Adapted from Wang et al. (2013).

Table 1: Linguistically-motivated rules to remove constituents in sentence compression. The elements in brackets are removed.

3.2 Content Realization

Our system implements sentence cleaning and compression by pre-processing sentences to produce multiple compressed candidates for the content selection module, following the approach used in Zajic et al. (2007), Vanderwende et al. (2007), and Wang et al. (2013). Users have the option of choosing from seven linguistically-motivated rules to shorten candidate sentences as shown in Table 1.

3.2.1 Sentence Cleaning

We enable users the option to declutter raw input that detracts from readability. Nutshell allows users to generate candidate sentences with the removal of headers normally found in the first line of news articles (`remove_header`[1]), extraneous information found within parentheses (`remove_parens`), and unpaired quotes left in the raw input (`remove_quotes`). Lastly, independent of user selection, extra spaces and new lines are removed from each sentence. This cleaned version of the sentence is then added to the list of candidate sentences.

3.2.2 Sentence Compression

Nutshell uses spaCy, a Natural Language Processing open-source library for dependency parsing (Honnibal and Johnson, 2015). We chose spaCy due to its fast processing and high accuracy as evaluated in Choi et al. (2015). The spaCy dependency parser uses English dependency labels based on the ClearNLP project (Choi and Palmer, 2012). We chose spaCy's pre-trained statistical model en_core_web_md version 2.1.0, an English multi-task CNN trained on OntoNotes (AI, 2019).

After cleaning the sentence and obtaining a dependency parse, Nutshell navigates the parse tree of a sentence to find the user-selected grammatical constituents for removal. Users can remove noun appositives (`remove_appos`), adverbial clause modifiers (`remove_advcl`), relative clause modifiers (`remove_relcl`), and clausal modifiers of nouns (`remove_acl`). For each dependency that is specified for removal and is present in the sentence according to the parse, our system trims the node from the original cleaned version of the sentence, removes extraneous punctuation left after compression, and adds the new compressed sentence to the list of candidates for content selection. This can lead to up to four new candidates per sentence being added.

3.3 Content Selection

The Nutshell system selects sentences to build each topic summary from the cluster of topic documents via the Biased LexRank graph approach to determining sentence salience, which is based on a PageRank style of inter-sentential similarity weighting with an additional bias for the query topic (Otterbacher et al., 2005, 2009). In the Biased LexRank method, each node in the similarity graph represents a sentence in the topic document cluster (only sentences of `min_sent_len` or more words are included), and the edges between nodes represent the weight of their inter-sentential similarity and topic bias. This approach allows sentences to be ranked based on both their relation to the topic and to their centrality in the

[1] `monospace font` is used to denote parameters that may be set by the user to different values

topic document cluster.

3.3.1 tf-idf Language Model

tf Formulas: Nutshell uses tf (term frequency), building a language model of raw counts of word tokens for both sentences and entire topics. Nutshell also builds a normalized version of tf, allowing for the choice between two normalized tf-formulas. The first option for tf is called `term_frequency`, which normalizes by the total number of word tokens in a sentence (Manning et al., 2008):

$$tf_{w,s} = \frac{f_{w,s}}{|s|}$$

where $f_{w,s}$ is the count of words w in sentence s, and $|s|$ is the total number of all words in s.

The second option for tf is called `log_normalization`, which normalizes by assigning a weight motivated by the positive non-linear logarithmic scale. This is used for a large range of positive multiples and responds well to skew toward larger values:

$$tf_{w,s} = log(1 + f_{w,s})$$

idf Formulas: Nutshell builds an idf (inverse document frequency) model for the topic based on a standard formula, and additionally builds a modified version of idf, allowing for two options.

`standard_idf` provides a logarithmic scaled measure of how much information a word provides (Manning et al., 2008):

$$idf_w = log(\frac{N}{n_w}) = -log(\frac{n_w}{N})$$

where N is the number of sentences in the topic, and n_w is the number of sentences where word w occurs. Note that depending on the implementation division by zero is possible and must be dealt with accordingly.

The first idf option is `smooth_idf`, which, because of its continuous nature, provides a measure of word relevance for any value of input and removes the need to prevent zero division:

$$idf_w = log(\frac{N}{1 + n_w})$$

The second option is `probabilistic_idf`, which is derived from estimating the probability that a given document contains a particular term:

$$idf_w = log(\frac{N - n_w}{n_w})$$

tf-idf Formula: Two tf-idf language models are built, one using raw counts, and the second using whichever normalization options are chosen for tf and idf. The calculation for tf-idf for any word token in a sentence is therefore:

$$\text{tf-idf}_{w,s} = tf_{w,s} \times idf_w$$

3.3.2 Biased LexRank Graph Approach

Nutshell uses the Biased LexRank graph approach as described in Otterbacher et al. (2005) and Otterbacher et al. (2009), which uses the generalized formula:

$$BLR(s|q) = d\frac{b(s|q)}{\sum_{z \in C} b(z|q)} + (1-d) \sum_{v \in C} \frac{w(v,s)}{\sum_{z \in C} w(z,v)}$$

where $BLR(s|q)$ gives the Biased LexRank score of a sentence s given query topic q, d is the weight to give to the topic bias (higher values of d favor the bias over the inter-sentential similarity), b is the topic bias weight for s given q, and w is the inter-sentential similarity weight for s and all adjacent sentences v in the topic document cluster C.

Nutshell allows the user to choose between various formulas for the calculation of the bias b and the inter-sentential similarity w, as well as to set the value of d. The bias can be set to `cos` (cosine similarity), `rel` (relevance), or `gen` (generative probability). The inter-sentential similarity can set to `cos` (cosine similarity) or `norm` (normalized generative probability).

Cosine Similarity: Nutshell uses the tf-idf modified version of cosine similarity from Otterbacher et al. (2005), where the similarity between two sentences, x and y, is defined as:

$$\text{cos\_sim}(x,y) = \frac{\sum_{w \in x,y} (\text{tf-idf}_{w,x}) \times (\text{tf-idf}_{w,y})}{\sqrt{\sum_{x_i \in x} (\text{tf-idf}_{x_i,x})^2} \times \sqrt{\sum_{y_i \in y} (\text{tf-idf}_{y_i,y})^2}}$$

For inter-sentential cosine similarity, Nutshell uses a minimum threshold `intersent_threshold` to ensure that edge weights are only added when the minimum amount of similarity is found in order to reduce possible noise.

Relevance: An alternative method for similarity weighting for the bias is relevance (Otterbacher

et al., 2005), which calculates the similarity between a sentence s and a query topic q based on the tf and idf for word w:

$$\text{rel}(s|q) = \sum_{w \in q} log(\text{tf}_{w,s} + 1) \times log(\text{tf}_{w,q} + 1) \times \text{idf}_w$$

(Normalized) Generative Probability: In Otterbacher et al. (2009), another method for similarity was introduced based on the generative probability of a sentence u given another sentence v, using a tf language model. This method is comprised of a number of subformulas.

First, there is a subformula for $\text{P}_{ML}(w|v)$, which is the Maximum Likelihood Estimate (MLE) probability of word w given a sentence v:

$$P_{ML}(w|v) = \frac{tf_{w,v}}{|v|}$$

This can then be smoothed for unseen words with the Jelinek-Mercer (JM) method (Otterbacher et al., 2009), using P_{ML} values for both the sentence v and the entire cluster of sentences in the topic C, with a λ value determining the weight given to the cluster:

$$P_{JM}(w|v) = (1 - \lambda)P_{ML}(w|v) + \lambda P_{ML}(w|C)$$

Using this smoothed MLE probability, we can calculate the generative probability for sentence u given sentence v:

$$P_{gen}(u|v) = \prod_{w \in u} P_{JM}(w|v)^{tf_{w,u}}$$

This generative probability can optionally be normalized by sentence length as well:

$$P_{norm}(u|v) = \left(\prod_{w \in u} P_{JM}(w|v)^{tf_{w,u}} \right)^{\frac{1}{|u|}}$$

3.3.3 Power Iteration Method

The Biased LexRank formula, using the chosen bias and inter-sentential weighting formulas, can be translated into a Markov Matrix, where A is the bias vector of similarity between sentences and the topic and B is the inter-sentential similarity matrix:

$$M = [dA + (1 - d)B]$$

With this Markov Matrix, we implement the power iteration method to find the final Biased LexRank values for each sentence. Beginning with p_0 as a uniform probability distribution, we iterate over the following formula until convergence (where the average change in probabilities between iterations in less than `epsilon`), updating p at each step:

$$p_t = M^T p_{t-1}$$

This process gives a final set of scores, p_t for ranking sentence salience.

3.3.4 Sentence Selection

Nutshell uses a greedy iterative approach to choose sentences for the summary. Given a set of sentences sorted by their Biased LexRank scores, we iteratively select the highest-ranked sentence that fits within the 100 word requirement. We filter out sentences with a cosine similarity greater than or equal to a `summary_threshold` with any of the other selected sentences to eliminate potential redundancy.

3.4 Information Ordering

Nutshell offers users two approaches for information ordering: Chronological Ordering (`chron`) and Entity-Based Ordering (`entity`).

3.4.1 Chronological Ordering

Our system implements a naïve ordering algorithm, Chronological Ordering, which uses the date of each document to order sentences, as described in Barzilay et al. (2002). Additionally, sentences are ordered by their original position in the document for any sentences from the same document.

3.4.2 Entity Grid Based Ordering

Nutshell implements the Entity-Based Ordering approach as described in Barzilay and Lapata (2008). This approach models local coherence from entity-grid representations of discourse, which captures important probabilistic patterns of sentence transitions for entity mentions. We then use the entity-grid representations in a ranking-based SVM generation task. With the learned model, sentences are optimally ordered in a given document.

3.4.3 Entity Grids as Feature Vectors

Using the entity grid model from Barzilay and Lapata (2008), we represent all the nouns found in a summary as the columns of the grid and each sentence in the summary as a row, with the presence (X) or absence (–) of an entity in the sentence

as the cell value. We expect that most columns will be sparse, but those that are dense represent relevant subjects or objects that can contribute to having a high-coherence text (Barzilay and Lapata, 2008).

We can use this grid to create a vector of local entity transitions between sentences, which is a sequence of $\{X, -\}^n$. We use transition sequences of length two (e.g. [X –]) to compute the transition probabilities by getting the ratio of the sequence probability divided by the total number of transitions. Thus, each document is distributed over entity transition types. Transition probabilities can be represented using standard feature vector notation. According to Barzilay and Lapata (2008), every permutation *j* of the grid of a document d_i can be represented by a feature vector:

$$\phi(x_{ij}) = (p_1(x_{ij}), p_2((x_{ij}), ..., p_m(x_{ij}))$$

where *m* is the number of all predefined entity transitions, and $p_t(x_{ij})$ the probability of transition *t* in grid x_{ij}.

This feature vector representation allows us to use machine learning algorithms, particularly in text generation, where instances should be ranked rather than classified. Like Barzilay and Lapata (2008), we treat our task as a Support Vector Machine (SVM) optimization problem where we learn a ranking function to get the optimal sentence ordering in a given summary.

3.4.4 Grid Construction: Linguistic Dimensions

We replicate the basic version of how entity grid features are modeled in Barzilay and Lapata (2008).

Entity Extraction: Nutshell implements a simplified version of coreference resolution, using the NLTK POS-tagger (Loper and Bird, 2002) to capture nouns and treats each as an entity which is coreferent to another noun only if it is exactly identical to it.

Grammatical Function: Barzilay and Lapata (2008) use four categories to express transitions in robust grids in the set: $\{S, O, X, -\}$ and two categories in simplified grids: present (X) or absent (–) in a sentence. We generate simplified grids with the latter technique. Since we use binary category types, we do not employ a parser for grammatical function. Given a set of all nouns in a document, a noun is either recorded as present or absent for each sentence in the document.

Salience: Barzilay and Lapata (2008) identify salient entities based on their frequency. We employ the same technique where given a document, we compute the transition probabilities for each salience group, and then combine them into a single vector. Given *n* transitions with *k* salience classes, the feature space will be of size $n \times k$. We also use the same binary distinction for salience classes where $k = 2$.

3.4.5 SVM Model

Nutshell uses Joachims'(2006) SVM^{rank} package with all parameters set to default values to train and output a model which can determine the best ordering of sentences.

Training: The training data is comprised of human-created gold-standard model summary files from news articles in TAC 2009. These gold-standard sentence orderings are then permuted a variable number of times (`num_permutations`), creating negative training examples. We feed the feature training set into an SVM to determine the hyperplane that differentiates between gold (positive) and permuted (negative) orderings.

Testing: The testing data is comprised of permutations of sentences in each topic from the content selection module. After learning the ideal ranking function, the model can then be used to determine the best sentence ordering for unlabeled test vectors. We select the highest ranked ordering, and use this sentence ordering for the final summary.

4 Results

In order to evaluate the performance of the Nutshell system, we ran it on the Text Analysis Conferences (TAC) shared evaluation tasks, which are topic-focused multi-document text summarization tasks of news article document clusters from ACQUAINT, ACQUAINT2, and Gigaword with human-generated gold model summaries. We used the TAC 2010 data for development testing (devtest), and the TAC 2011 data for evaluation testing (evaltest). We used two metrics for evaluation, ROUGE-1 (unigram) average recall and ROUGE-2 (bigram) average recall.

Below we present the ROUGE results of tuning our Nutshell D3 system on the TAC 2010 devtest data, the ROUGE results of running our Nutshell D4 system on both the TAC 2010 devtest and

TAC 2011 evaltest data, a comparison of Information Ordering approaches, a comparison of Content Realization sentence cleaning and compression techniques, and an exploration of readability errors.

4.1 Nutshell D3 Tuning

The Nutshell D3 system, which lacked the Content Realization module, was tuned on the TAC 2010 devtest data to determine the best Biased LexRank formula configurations, the best hyperparameter values, and the best Information Ordering approach.

4.1.1 Biased LexRank Configurations

We ran our D3 system using three configurations of Biased LexRank to compare the performance of the system based on various weighting formulas.

D2 Configuration (`cos, cos`): The D2 Configuration is the configuration we used in another previous version of the nutshell system, which used cosine similarity for both the bias and inter-sentential weighting:

$$\text{BLR}(s|q) = d\frac{\text{cos\_sim}(s,q)}{\sum_{z\in C}\text{cos\_sim}(z,q)} + (1-d)\sum_{v\in C}\frac{\text{cos\_sim}(s,v)}{\sum_{z\in C}\text{cos\_sim}(z,v)}$$

2005 Configuration (`rel, cos`): The 2005 Configuration is based on Otterbacher et al. (2005), which used relevance for the bias and cosine similarity for inter-sentential weighting:

$$\text{BLR}(s|q) = d\frac{\text{rel}(s|q)}{\sum_{z\in C}\text{rel}(z|q)} + (1-d)\sum_{v\in C}\frac{\text{cos\_sim}(s,v)}{\sum_{z\in C}\text{cos\_sim}(z,v)}$$

2009 Configuration (`gen, norm`): The 2009 Configuration is based on Otterbacher et al. (2009), which used generative probability for the bias and normalized generative probability for the inter-sentential weighting:

$$\text{BLR}(s|q) = d\frac{\text{gen}(q|s)}{\sum_{z\in C}\text{gen}(q|z)} + (1-d)\sum_{v\in C}\frac{\text{norm}(v|s)}{\sum_{z\in C}\text{norm}(v|z)}$$

4.1.2 Hyperparameters

The Nutshell D3 system involves 13 hyperparameters that needed to be tuned to work with the TAC task, including hyperparameters for Data Preparation (which also builds the tf-idf language model), for Content Selection Biased LexRank formulas, and for the Information Ordering component. For each of the three configurations, all hyperparameters were optimized for ROUGE-2 (R-2) scores one at a time by holding the other hyperparameters constant, and then updating and fixing the value of the newly tuned hyperparameter before tuning the next.

The default baseline hyperparameters that tuning began with were the same as in D2, with the addition of the new hyperparameters added to the program whose defaults were that used in the corresponding papers (Otterbacher et al., 2005, 2009). These are shown in Table 2 along with the baseline R-2 scores. "–" is used for hyperparameters that are irrelevant for a particular configuration. All hyperparameter tuning used chronological Information Ordering except for `num_permutations`, which is a hyperparameter for entity grid based ordering.

Hyperparam	D2	2005	2009
stemming	False	False	False
lower	False	False	False
tf	term_freq	term_freq	term_freq
idf	smooth	smooth	smooth
d	0.7	0.7	0.7
summary_thresh	0.5	0.5	0.5
epsilon	0.1	0.1	0.1
min_sent_len	5	5	5
include_narr	False	False	–
intersent_thresh	0.0	0.0	–
mle_lambda	–	–	0.6
k	–	–	20
num_perm	–	–	–
R-2 score	0.05600	0.05950	0.05390

Table 2: Baseline hyperparameter settings and ROUGE-2 average recall scores on TAC 2010 for each Nutshell configuration.

The following sections present the R-2 score improvements as each hyperparameter was tuned, with the tuned values for each hyperparameter presented at the end.

Data Preparation Hyperparameters: For data preparation, four hyperparameters were tuned, as is shown in Table 3.

As can be seen in the results, the first step of tuning `stemming` by changing the default value had a large effect, improving the R-2 score by more than 0.01 for each configuration. The defaults for `lower` and `tf_type` were already the best values for these two hyperparameters, leading to no further gains. For `idf_type`, there were very minor gains for the D2 and 2005 mod-

Hyperparam	D2	2005	2009
stemming	0.07019	0.07040	0.06704
lower	0.07019	0.07040	0.06704
tf	0.07019	0.07040	0.06704
idf	0.07100	0.07088	0.06704

Table 3: ROUGE-2 average recall scores on TAC 2010 for tuned Data Preparation hyperparameters.

els when it was set to `standard_idf`. Given that this value change for `idf_type` was so small and that it also negatively impacted the 2009 configuration more (lowering its R-2 from 0.06704 to 0.06592), the default was kept for this hyperparameter in order to have a single set of hyperparameter values for model training to reduce run time for each tuning run.

Content Selection Hyperparameters: In the same manner as for Data Preparation, the hyperparameters for the Biased LexRank formulas in Content Selection were tuned one at a time, with the tuned values at each step being fixed before tuning the next hyperparameter.

Hyperparam	D2	2005	2009
d	0.07755	0.0743	0.07361
summary_thresh	0.0784	0.07568	0.07444
epsilon	0.07868	0.07592	0.07444
min_sent_len	0.07878	0.07601	0.07454
include_narr	0.07878	0.07601	–
intersent_thresh	0.07878	0.07827	–
mle_lambda	–	–	0.07454
k	–	–	0.07622

Table 4: ROUGE-2 average recall scores on TAC 2010 for tuned Content Selection hyperparameters.

The biggest gains were seen in the change to the first hyperparameter, `d`, with an R-2 score increase of about 0.003-0.007 for each configuration, as can be seen in Table 4. Each subsequent hyperparameter tuning improved performance by a small amount (< 0.002) or showed no change due to the default being the best value.

Information Ordering Hyperparameters: There was a single hyperparameter that needed to be tuned for Information Ordering, `num_permutations`, which additionally required `info_order_type` to be set to `entity`.

Table 5 shows that using entity grid based ordering and tuning this hyperparameter slightly increased the R-2 score for the D2 and 2005 configurations, although it lowered the score for the 2009 configuration.

Hyperparam	D2	2005	2009
num_perm	0.07891	0.08017	0.07592

Table 5: ROUGE-2 average recall scores on TAC 2010 for tuned Information Ordering hyperparameter.

Tuned Hyperparameter Values: Table 6 shows the hyperparameter values that each configuration was tuned to in order to reach the tuned R-2 scores.

Hyperparam	D2	2005	2009
stemming	True	True	True
lower	False	False	False
tf	term_freq	term_freq	term_freq
idf	smooth	smooth	smooth
d	0.1	0.2	0.3
summary_thresh	0.4	0.3	0.4
epsilon	0.02	0.04	0.1
min_sent_len	3	3	4
include_narr	False	False	–
intersent_thresh	0.0	0.1	–
mle_lambda	–	–	0.6
k	–	–	9
num_perm	5	5	–
R-2 score	0.07891	0.08017	0.07622

Table 6: Tuned hyperparameter settings and ROUGE-2 average recall scores on TAC 2010 for each Nutshell configuration.

4.1.3 Information Ordering Approach

Using the tuned hyperparameters discussed above, we ran Nutshell using the three configurations (D2, 2005, 2009) for content selection and the two options (`chron`, `entity`) for the newly introduced Information Ordering component to determine which version of Nutshell performed the best on the TAC 2010 task. Table 7 shows that entity based ordering for the D2 configuration had the highest R-1 score, and entity based ordering for the 2005 configuration had the highest R-2 score.

4.2 Nutshell D4

Nutshell D4 is an extension of Nutshell D3 with the addition of a Content Realization component.

	R-1	R-2
Nutshell D2 (chron)	**0.27618**	0.07878
Nutshell D2 (entity)	0.27486	0.07891
Nutshell 2005 (chron)	0.27454	0.07827
Nutshell 2005 (entity)	0.27487	**0.08017**
Nutshell 2009 (chron)	0.2612	0.07622
Nutshell 2009 (entity)	0.25925	0.07592

Table 7: ROUGE average recall scores on TAC 2010 for Nutshell configurations with chronological versus entity information ordering.

For Nusthell D4, the D3 Baseline uses the Nutshell D3 2005 (entity) configuration, but has slightly lower ROUGE scores (R-2 is 0.07661 instead of 0.08017) due to some changes in the preprocessing of data. This baseline is then compared to the D4 system as each Content Realization cleaning and compression technique is used to produce a new version of the sentences. Table 8 presents the results for the TAC 2010 devtest data, and Table 9 presents the results for the TAC 2011 evaltest data. In each table, each line represents the cleaning and parsing of all parameters up to and including that line. For example, "D4 -parenthetical" represents the Nutshell D4 system using the same parameters as the D3 Baseline system with the additional removal of headers and parenthetical information.

	R-1	R-2
D3 Baseline	0.27132	0.07661
D4 -header	0.26685	0.07201
D4 -parenthetical	0.26486	0.07225
D4 -unpaired quotes	0.26449	0.07195
D4 -appositional mod	0.26321	0.07272
D4 -adverbial clause mod	0.26704	0.07466
D4 -relative clause mod	**0.27223**	0.07725
D4 -clausal mod	0.27142	**0.07766**

Table 8: ROUGE average recall scores on TAC 2010 devtest for Nutshell D4 with various content realization cleaning and compression settings.

Both Table 8 and Table 9 show that the ROUGE scores, especially the R-2 scores, decrease after cleaning the sentences by removing headers, parenthetical information, and unpaired quotes (despite the significant increase to readability these offer), but increase again after some of the sentence compression parameters are applied. For devtest (Table 8), the highest R-2 score (0.07766) is

	R-1	R-2
D3 Baseline	0.29244	0.08537
D4 -header	0.29234	0.08537
D4 -parenthetical	0.29099	0.08436
D4 -unpaired quotes	0.28894	0.08332
D4 -appositional mod	0.29377	0.08689
D4 -adverbial clause mod	**0.29671**	**0.08787**
D4 -relative clause mod	0.29417	0.08572
D4 -clausal mod	0.29326	0.08742

Table 9: ROUGE average recall scores on TAC 2011 evaltest for Nutshell D4 with various content realization cleaning and compression settings.

when all cleaning and compression parameters are applied. For evaltest (Table 9), the highest R-2 score (0.08787) is after the adverbial clause modifiers are removed, although the R-2 score when all parameters are applied is very close (0.08742). Due to this, we chose to use the version of D4 with all sentence cleaning and compression parameters applied as our final system configuration. Note that this does significantly increase the amount of time it takes to run the Nutshell system, from around 5-10 minutes to run both devtest and evaltest on the D3 configuration to around 20-45 minutes to run both on the D4 final configuration.

Using this final Nutshell D4 system configuration, we can additionally compare against the LEAD and MEAD baselines for TAC 2010 and TAC 2011. Table 10 shows that the Nutshell D4 system significantly outperformed the R-2 scores (by around 0.02) of the LEAD baseline for TAC 2010 and TAC 2011, as well as the MEAD baseline for TAC 2010. It slightly outperformed the R-2 score (by 0.0006) for the MEAD baseline for TAC 2011.

	2010 R-2	2011 R-2
LEAD baseline	0.05376	0.06410
MEAD baseline	0.05927	0.08682
Nutshell D4	**0.07766**	**0.08742**

Table 10: ROUGE average recall scores on TAC 2010 and TAC 2011 for different baseline systems and Nutshell D4

4.3 Information Ordering Comparison

Overall, entity grid based Information Ordering had a positive effect on the order in which information was presented in many of the summaries.

Many of the entity summaries presented more general information and introductions to topics first, whereas the chronological ordering presented details first.

For example, entity ordering produced the following summary, where purging disorder is introduced as a topic before adding details:

Just as binge eating disorder has many of the characteristics of bulimia, so does purging disorder. Some psychiatrists want to create a different label for EDNOS, calling it instead "mixed eating disorder" or "atypical eating disorder." The other group had no prior history of eating disorders, but a life event – a divorce, loss of parent or job – triggered an eating disorder. "When a person has an eating disorder, it becomes a family problem," Hecht says. "I know I need help. Medical professionals say they see more and more adult women like Hecht – that is, those older than college age – suffering from eating disorders.

In contrast, the chronological ordering of this summary begins with details about a specific woman suffering from an eating disorder:

Medical professionals say they see more and more adult women like Hecht – that is, those older than college age – suffering from eating disorders. "When a person has an eating disorder, it becomes a family problem," Hecht says. Just as binge eating disorder has many of the characteristics of bulimia, so does purging disorder. Some psychiatrists want to create a different label for EDNOS, calling it instead "mixed eating disorder" or "atypical eating disorder." "I know I need help. The other group had no prior history of eating disorders, but a life event – a divorce, loss of parent or job – triggered an eating disorder.

The system does not perfectly produce well-ordered sentences for summaries, but it does offer a slight improvement in many cases.

4.4 Content Realization Comparison

Nutshell's Content Realization module allows for sentence cleaning, which can improve readability, and sentence compression, which can improve the content coverage of the summary. By removing unnecessary content, these also allow the limited word-count in each summary to be used on more important information. Below we look at the changes made to summaries when these techniques are applied.

4.4.1 Sentence Cleaning

Sentence cleaning, which includes removing headers, parenthetical information, and unpaired quotation marks, can improve the readability of sentences by removing distracting punctuation errors and irrelevant information, as well as allowing the Nutshell system extra room to fit in more content.

For example, before sentence cleaning was applied, Nutshell produced the following summary, which included multiple headers and had unpaired quotes:

PORT MORESBY, Papua New Guinea (AP) _ A tsunami spawned by a 7.0 magnitude earthquake crashed into Papua New Guinea's north coast, crushing villages and leaving hundreds missing, officials said Sunday. Authorities at Aitape in the West Sepik province, on Papua New Guinea's northwest coast, said the tsunami that hit the coast west of the village on Friday night had wiped out three villages and had almost completely destroyed another. "They're dead . CANBERRA, July 18 (Xinhua) – Australia will provide transport for relief supplies and a mobile hospital to Papua New Guinea (PNG) following Friday's tsunami tragedy.

After cleaning, this summary reads much more like regular prose:

Authorities at Aitape in the West Sepik province, on Papua New Guinea's northwest coast, said the tsunami that hit the coast west of the village on Friday night had wiped out three villages and had almost completely destroyed another. A tsunami spawned by a 7.0 magnitude earthquake crashed into Papua New Guinea's north coast, crushing villages and leaving hundreds missing, officials said Sunday. The death toll in Papua New Guinea's tsunami disaster has climbed to 599 and is expected to rise, a PNG disaster control officer said Sunday. They're dead . It's complete devastation. The beaches are clean, Cassey said.

4.4.2 Sentence Compression

Nutshell uses sentence compression to remove syntactic clauses (noun appositives, adverbial clause modifiers, relative clause modifiers, and clausal modifiers of nouns) to produce multiple alternative candidates of an original sentence from which the Content Selection module may choose. This can improve the density of information as shorter versions of sentences with irrelevant clauses removed are available for selection. Note, however, that it can also lead to readability errors, as is covered in section 4.5.

An example of a more dense version of a sentence is below, where the sentence was shortened via compression by the removal of the section in brackets, while still retaining its primary message:

*Representatives, including the commercial fishing industry, government regulators and environmental groups, plan **[to discuss ways to strengthen information sharing and cooperation among regional organizations]** to better manage tuna stock and adopt an action plan, the Japanese Fisheries Agency said in a statement.*

By providing multiple shorter, compressed versions of sentences, the Content Selection module frequently selected different sets of sentences for the summaries. For example, before the compressed sentences were added, one summary read:

*Seven near-simultaneous bomb blasts tore through crowded markets in the Indian tourist city of Jaipur Tuesday, killing at least 80 people and wounding 200 in what police said was a terror attack. Seven bombs ripped through the city of Jaipur, leaving at least 60 people dead. **It's a terror attack.** In the first terrorist attack in many months, seven bombs went off within minutes of one another on Tuesday evening in the crowded lanes of one of India's main tourist hubs, the historic city of Jaipur, with reports of deaths **[ranging from 50 upward]**, with roughly 150 injured, officials said.*

After adding the compressed sentences, the summary became:

*Seven near-simultaneous bomb blasts tore through crowded markets in the Indian tourist city of Jaipur Tuesday, killing at least 80 people and wounding 200 in what police said was a terror attack. Seven bombs ripped through the city of Jaipur, leaving at least 60 people dead. **Pakistan denies any role in the bombings.** In the first terrorist attack in many months, seven bombs went off within minutes of one another on Tuesday evening in the crowded lanes of one of India's main tourist hubs, the historic city of Jaipur, with reports of deaths, with roughly 150 injured, officials said.*

In this example, the removal of *"ranging from 50 upward"* (in brackets above) allowed enough extra room in the summary for the sentence *"It's a terror attack"*, which was repetitive information, to be replaced by *"Pakistan denies any role in the bombings"*, which provided new, additional content.

4.5 Readability

It is to be expected that primarily extractive summaries will contain a variety of errors regarding entity mentions and clausal level violations (Friedrich et al., 2014), and that compression itself can lead to errors due to its abstractive nature. Below we discuss the errors that Nutshell produces and readability evaluations of its summaries.

4.5.1 Entity Mention Violations

One of the most common types of readability errors that Nutshell makes is in regards to entity mentions. In particular, it often will use pronouns, definite articles, and last names only in the first mention of an entity:

*Drought changed all **that** ...*
***They** ascribed ...*
*The letter was from Sudan's foreign minister, **Erwa** said.*

Alternatively, Nutshell will also commonly use the full name of an entity or explanation in subsequent mentions:

Debra Lafave, the former Tampa middle school teacher accused of having sex with a 14-year-old male student ...Debra LaFave had sex with a 14-year-old student

4.5.2 Clausal Level Violations

Although Nutshell improved readability in D4 by removing extraneous headers and quotation marks, not all of these instances were caught due to inconsistent formatting that did not follow typical patterns:

Cox News Service RICHMOND, Va. ?-

There are also a number of discourse relation errors that Nutshell does not account for:

***But** in Sichuan and in Shaanxi province, ...*
***By comparison**, Floyd's hurricane-force wind ...*

Redundancy continues to plague summaries as well, where the same information is repeated more than once:

on Friday night ... on Friday night

Lastly, Nutshell itself introduces grammatical violations and leads to nonsense sentences due to clausal removal where it is not optional

information:

They, he said.
And it has China, again.
Chen also stressed that.

4.5.3 Readability Evaluations

A simple human readability evaluation was conducted for Nutshell D4 and nine other peer systems for the TAC 2011 task. Table 11 shows the rating scale (1-5) and associated description used to evaluate the topic summaries for readability.

Rating	Description
1	**Poor**: confusing, bizarre, or incoherent; unrecoverable contexts
2	**Fair**: several readability problems, but most are recoverable
3	**OK**: mostly understandable; a few minor readability or flow issues
4	**Good**: fully coherent; minor stylistic problems
5	**Excellent**: idiomatically correct, readable, and cohesive

Table 11: Readability rating scale and description for TAC topic summaries

For each system, four readers evaluated a selection of topic summaries using this rating scale and their ratings were averaged to obtain a single rating value for each topic. Table 12 shows Nutshell D4's ratings for the six topics that were evaluated, as well as the across-team mean and standard deviation (stddev) for those topics. Nutshell D4 had an average rating of 3.069 for its summaries, which was comparable to the across-team average of 3.067 and shows that the summaries produced are mostly understandable despite minor errors.

Topic	Nutshell D4	Mean	Stddev
D1105A	2.7	3.2	0.7
D1106A	3.0	3.0	0.8
D1107B	3.3	3.0	0.8
D1108B	3.0	2.9	0.94
D1109B	3.3	2.9	0.6
D1110B	3.3	3.4	0.7
Average	3.069	3.067	0.757

Table 12: Readability scores for selected topics in TAC 2011 for Nutshell D4 with across-team mean and stddev

5 Discussion

Nutshell uses a Biased LexRank similarity graph model, which allows us to account for both the relevance of sentences within a document cluster as well as their connections to the topic itself. Although multiple versions of similarity weighting are possible with Nutshell, the best performance was achieved through using a relevance formula for the topic bias and cosine similarity for inter-sentential similarity. To improve the language model that these similarity calculations are based on, we used the Porter Stemmer and removed stopwords as part of the data preprocessing.

We tested both chronological and entity-based ordering of extracted sentences as ways to improve R-2 scores. We found that entity-based ordering produced higher-scoring summaries for the D2 and 2005 configurations. A manual analysis indicates that these orderings tend to present general information first, then elaborate into details.

Instead of relying on the settings used in the original LexRank papers (Erkan and Radev, 2004; Otterbacher et al., 2005, 2009), we tuned the hyperparameters for our system, such as the formulas used for calculating tf-idf and sentential similarity, on the TAC 2010 devtest data to determine what would produce the best results. Our tests found that entity-based ordering for the 2005 configuration produced the highest R-2 scores.

In our D4 system, to improve readability and increase content coverage as a part of Content Realization, we implemented a number of strategies documented by linguists as valuable for sentence compression. We remove headers, parenthetical information, unpaired quotes, and a variety of clausal modifiers. This allows more information to be included and higher-ranked sentences can be selected because of the space freed up from eliminating unnecessary information. We see R-2 scores dip after sentence cleaning, but then increase to above our baseline scores after compression is applied, indicating that the information contained in the clauses is not generally as valuable to ROUGE scores as information from extra sentences that can be included within the same number of words. One explanation for unexplained decreases caused by some parameters could be due to tie-breaking in our ordering algorithm. Summaries that were otherwise identical had different orderings for each of these tests, and without controlling for order, it is difficult to tell

what is indeed causing the score changes. While in coordination, these strategies do increase scores to a degree, they introduce new readability issues caused by the challenging nature of stepping into a more abstractive method of summarization, although Nutshell performs comparably to peer systems on readability evaluations.

Overall, Nutshell performs topic-focused extractive text summarization well, outperforming the LEAD an MEAD baseline R-2 scores by about 0.02 (about a 35% increase) on TAC 2010 and the LEAD baseline on TAC 2011, and slightly outperforming the MEAD baseline by 0.006 on TAC 2011.

5.1 Future Work

The continued improvement of Content Realization sub-components would likely lead to higher scores and better readability. In future work, we might continue to hone techniques for removing metadata and punctuation-based patterns, which may be less valuable in summaries than other data. We also could experiment with coreference resolution and the use of named entity recognition to shorten entity references.

The integration of machine learning into the selection of which parameters correlate most highly with ROUGE scores could also provide a more effective system than our current rule-based strategy.

6 Conclusion

Nutshell is a topic-focused multi-document extractive text summarization system. Nutshell pre-processes data by tokenizing sentences, removing stopwords, and optionally lower-casing and stemming tokens, to build a tf-idf language model based on words in sentences in a topic document cluster. Sentences are cleaned and compressed using pattern-based rules so that certain unnecessary information (such as certain types of clauses) is optionally removed from sentences. Nutshell implements content selection using the power iteration method and the Biased LexRank similarity graph approach to sentence salience, which calculates weights for both inter-sentential similarity and topic-sentence bias using one of the formula options (cosine similarity, relevance, generative probability, or normalized generative probability). Summaries are built using the scores produced by this method in a greedy iteration approach that incorporates the 100 word limit and redundancy constraints. Users can choose to order sentences chronologically according to their original document publication date, or to make use of an entity-grid method which detects feature patterns of gold-standard orderings and selects an optimal order from permutations of sentences. The summaries that Nutshell produces outperform the LEAD and MEAD baseline in TAC 2010 and TAC 2011 ROUGE-2 metrics.

References

Explosion AI. 2019. encorewebmd: English multi-task cnn trained on ontonotes.

Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Jinho D Choi and Martha Palmer. 2012. Guidelines for the clear style constituent to dependency conversion. *Technical Report 01–12*.

Jinho D. Choi, Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. 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)*, pages 387–396, Beijing, China. Association for Computational Linguistics.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

Annemarie Friedrich, Marina Valeeva, and Alexis Palmer. 2014. LQVSumm: A corpus of linguistic quality violations in multi-document summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1591–1599, Reykjavik, Iceland. European Language Resources Association (ELRA).

David Graff. 2002. The aquaint corpus of english news text. Web Download. Philadelphia: Linguistic Data Consortium.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA. Association for Computing Machinery.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*. Philadelphia: Association for Computational Linguistics.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Vineeth G. Nair. 2014. *Getting Started with Beautiful Soup*. Packt Publishing.

Jahna Otterbacher, Gunes Erkan, and Dragomir Radev. 2005. Using random walks for question-focused sentence retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 915–922, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Jahna Otterbacher, Gunes Erkan, and Dragomir R. Radev. 2009. Biased lexrank: Passage retrieval using random walks with question-based priors. *Inf. Process. Manage.*, 45(1):42–54.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition. Web Download. Philadelphia: Linguistic Data Consortium.

MF Porter. 1980. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing and Management*, 43(6):1606–1618.

Ellen Vorhees and David Graff. 2008. Aquaint-2 information-retrieval text research collection. Web Download. Philadelphia: Linguistic Data Consortium.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1384–1394. Association for Computational Linguistics.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management*, 43(6):1549–1570.