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, which 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 baselines, as well an earlier (D2) untuned version of Nutshell which used chronological ordering, by over 35% on ROUGE-2 scores.

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 and ACQUAINT2, and produces 100-word extractive summaries for each topic cluster for the TAC 2010 task. Nutshell makes use of 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 require hyperparameters, 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 Text Analysis Conferences (TAC) shared task evaluation using ROUGE-1 and ROUGE-2 metrics. We find that Nutshell outperforms the LEAD and MEAD baselines, as well as a previous (D2) untuned base version of Nutshell, by about 35%.

2 System Overview

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

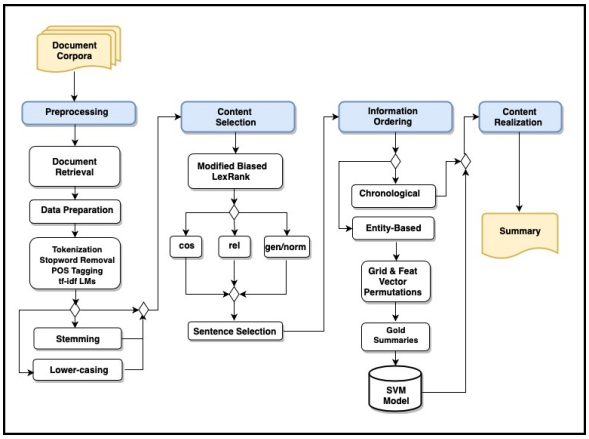

Figure 1: System Architecture

First, using the input document clusters, it preprocesses the XML files. Then, the data is parsed for tokenization, stop-word removal, and POS tag-

ging. It creates the 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 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, it 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 orders 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 user can select the number of sentence order permutations for training and testing. The sentence ordering with the highest model score ranking is selected to use for the summary.

Finally, the summary undergoes content realization, which currently does not make any changes, and then is directly output and evaluated using ROUGE metrics.

3 Approach

Below, we describe our approach in four subcomponents: data preparation, content selection, information ordering, and content realization.

3.1 Data preparation

For the TAC 2010 task, we collect data from a set of NIST standard XML files that identify a set of 44 training topics. These topics are parsed using the Beautiful Soup Python library (Nair, 2014). Furthermore, NLTK is used (Loper and Bird, 2002) 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.

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 and make necessary modifications to adhere to file path naming conventions.

3.2 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`[1] 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 topic document cluster.

3.2.1 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):

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

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

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:

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

3.2.2 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, it 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 $P_{ML}(w|v)$, 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:

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

3.2.3 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.

3.2.4 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.

3.3 Information Ordering

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

3.3.1 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.

3.3.2 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.

3.3.3 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 most optimal sentence ordering in a given summary.

3.3.4 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 $k = 2$.

3.3.5 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 output sentences accordingly.

3.4 Content Realization

The current implementation of the Nutshell system copies the ordered sentences for each topic from the Information Ordering sub-component.

4 Results

In order to evaluate the performance of the Nutshell system, we ran it on the Text Analysis Conferences (TAC) 2010 shared evaluation task, which is a topic-focused multi-document text summarization task of news article document clusters from ACQUAINT and ACQUAINT2 with human-generated gold model summaries. We used two metrics for evaluation, ROUGE-1 (unigram) average recall and ROUGE-2 (bigram) average recall, and compared against the LEAD and MEAD baselines for ROUGE-2.

4.1 Biased LexRank Configurations

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

4.1.1 D2 Configuration: `cos, cos`

The D2 Configuration is the configuration we used in D2 using cosine similarity for both the bias and inter-sentential weighting:

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

4.1.2 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:

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

4.1.3 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:

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

4.2 Hyperparameter Tuning

The Nutshell system involves many hyperparameters that needed to be tuned to work with the TAC 2010 task, including hyperparameters for data preparation (which also builds the tf-idf language model), for content selection BLR 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 1 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.

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

4.2.1 Data Preparation Hyperparameters

For data preparation, four hyperparameters were tuned, as is shown in Table 2.

As can be seen, 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 models 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.

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

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 2: ROUGE-2 average recall scores on TAC 2010 for tuned data preparation hyperparameters.

4.2.2 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.

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 3. Each subsequent hyperparameter tuning improved performance by a small amount (< 0.002) or showed no change due to the default being the best value.

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 3: ROUGE-2 average recall scores on TAC 2010 for tuned content selection hyperparameters.

4.2.3 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`.

Hyperparam	D2	2005	2009
num_perm	0.07891	0.08017	0.07592

Table 4: ROUGE-2 average recall scores on TAC 2010 for tuned information ordering hyperparameter.

Table 4 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.

4.2.4 Tuned Hyperparameter Values

Table 5 shows the hyperparameter values that each configuration was tuned to in order to reach the tuned R-2 scores.

4.3 Systems ROUGE Comparison

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 6 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.

This version of Nutshell with the 2005 configuration and entity grid based information ordering significantly outperformed the previous untuned version of Nutshell from D2 as well as both the LEAD and MEAD baseline R-2 scores by over 0.02, as is shown in Table 7.

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

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

	R-1	R-2
LEAD baseline	–	0.05376
MEAD baseline	–	0.05927
Nutshell D2 (untuned)	0.22720	0.05710
Nutshell 2005 (entity)	**0.27487**	**0.08017**

Table 7: ROUGE average recall scores on TAC 2010 for different baseline systems and Nutshell 2005.

4.4 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.

4.5 Readability Errors

In addition to information ordering, content realization is an important component of a text summarization system, especially as pertains to readability errors. Currently, Nutshell does not implement any specific content realization measures to improve summaries beyond their original extractive content. This leads to a number of errors regarding entity mentions and clausal level violations, as is expected in extractive summaries (Friedrich et al., 2014).

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:

*Supporters of **the measure** ...*
***They** also found ...*
*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, the former Tampa teacher accused of seducing a teenage boy.

4.5.2 Clausal Level Violations

Some of the clausal level violations that Nutshell makes are due simply to formatting issues, such as having quotation marks that are only on one half of a sentence:

"What's the district attorney saying?

Others are due to including news metadata such as datelines and notes to the reader:

LITTLETON, Colo. (AP) – ...
OPTIONAL MATERIAL FOLLOWS.)

There are also a number of discourse relation errors:

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

5 Discussion

Overall, Nutshell performs topic-focused extractive text summarization well, outperforming the LEAD an MEAD baseline R-2 scores by over 0.02 (about a 35% increase).

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 perfor-

mance 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, 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, although there are a number of entity mention and clausal level violations that impair readability.

5.1 Future Work

Implementing more intentional versions of Content Realization sub-components will likely lead to better summaries and performance. In future work, we plan to add Named Entity Recognition components to both improve information ordering with a more sophistical entity grid, and to revise named entity mentions, such as using full name mentions first and pronouns subsequently, for better content realization. We will also remove news metadata, including datelines (ex. SYDNEY (AP) –), which are unnecessary in summaries. Finally, we will ensure quotation marks are matched appropriately.

6 Conclusion

Nutshell is a topic-focused multi-document extractive text summarization system. Nutshell preprocesses 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. 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 ROUGE-2 metrics. Modifications to the Nutshell system, such the integration of Named Entity Recognition for content realization, will likely improve performance and are areas of future work.

References

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.

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.

T. Joachims. 2006. Training linear svms in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

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.

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

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