Word Games: Coherent Document Reconstruction

Mitchell Douglass Caelin Tran
Stanford University
Department of Computer Science
{mrdoug95, cktt13}@stanford.edu

1. Introduction

The document reconstruction problem can be formalized as follows. Given a set of fragments S from document D, predict the most coherent sequence of fragments that resembles the original text. Therefore, our input is the set S of fragments, and our output is a sequence T of fragments.

Document reconstruction has several immediate applications. Given a collection of key sentences from related articles, an effective model for document reconstruction can generate a coherent summary of the topic, or, given a collection of candidate documents (e.g. news articles, student reports) such a model might provide insight into which is most coherent. A well-designed model for document reconstruction could also shed light on the broader, extremely challenging problem of modeling structure in and actually generating coherent natural language.

To make document reconstruction tractable, we try two separate approaches. Our first, the k-fragments approach, assumes that all documents are split into a constant number k of fragments of varying size. The rationale behind this is that each fragment position carries meaningful semantics. E.g., the first fragment might typically convey introductory information. This assumption breaks down with large k as documents may not share structure at that granularity. We use Naive Bayes, variants of K-means, and a KD tree as our models in this approach.

Our second, the transitional approach, assumes that all fragments are of a constant size (m sentences) and that the best way to form a coherent document from them is to chain fragments together so that adjacent text is as fluent as possible, i.e., transitions smoothly from fragment to fragment. We use logistic regression to predict the probability of pairwise fragment orderings (does fragment A follow B?) and beam search to find the global fragment ordering of maximum likelihood.

2. Related Work

Coherence is a property of natural language text that represents some measure of its connectivity or flow. For example, an incoherent text would be full of non sequiturs and would fail to convey meaning. Historically, the literature surrounding coherence in text has focused on using our understanding of language to provide the necessary structure to model coherence, such as the use of cosine similarity of the word vectors of adjacent sentences [2] or the transition probabilities of linguistic features [3] to order sentences.

However, approaches relying purely on handcrafted features have fallen by the wayside as data-driven approaches have risen in popularity and performance. Models that attempt to estimate local coherence, the probability that one sentence comes before another, using feature embeddings [4] or SVMs and discourse entities (coreferent nouns) [5] achieve decent performance on the text ordering problem. Our transitional approach also relies on embeddings to model local coherence. Namely, we represent fragments as sums of their word embeddings, capturing their overall meaning [7]. However, our approach addresses a larger problem: relating fragments of arbitrarily large documents rather than just single sentences of smaller texts.

More recently, models that focus on ultimately optimizing global coherence, the overall probability of the text, have become the gold standard. For example, Logeswaran et al.'s recurrent neural network system achieved a staggering 51.6% accuracy and $\tau = 0.72$ when reconstructing NIPS abstracts [1]. None of these models address text ordering the same way in which we tackle document reconstruction with the k-fragments approach, but excitingly our best model achieves 59.0% accuracy and $\tau = 0.814$ on documents about human settlements.

3. Dataset and Features

We use English-language dumps of the Wikipedia archive [9], a collection of roughly 4.3 million informative articles on any conceivable subject. We use a third-party

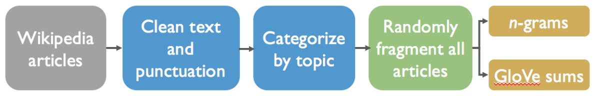

Figure 1: Dataset workup from raw to featurized input

tool called WikiExtractor [10] to remove all links, references, and extraneous formatting from the raw markdown, so that the cleaned articles consists only of lowercase English sentences. Using article metadata, we sort our articles into four common categories: people, films, settlements (cities, towns), and mixture (any topic). We produce training and test sets of articles from all categories, as well as for articles drawn from each category, by selecting 10,000 random articles from each set and partitioning them into train/test sets using the standard 80%/20% rule. As discussed in our methods section, we fragment these documents for training and testing using two distinct methods.

	Articles	Avg. Length (words)
Person Articles	120,000	600
Film Articles	75,000	595
Settlement Articles	160,000	562
All Categories	3,313,040	569

Here is an example article based on the Chinese mathematician Gu Deng:

> *Gu Deng (1882 - 1947) was a* ***mathematician*** *and* ***politician*** *at the end of Qing Dynasty. [...] Gu Deng was one of the* ***founders*** *of the Chinese* ***Mathematical*** *Society, the first* ***academic organization*** *for Chinese mathematicians,[...] After his* ***resignation****, he lived in obscurity, and is believed to have* ***died*** *around 1947.*

Note how the structure of the article can be captured with the semantics of key words. For example, personally descriptive terms like professions and titles, e.g., "founder", suggest that the person is being introduced in the first section. Terms regarding retirement and death suggest the last sentence is in the conclusion, providing evidence for our k-fragments positional semantics assumption. At the same time, words like "mathematician" link the first sentence to another containing "Mathematical" in the next paragraph, supporting our transitional approach.

To map raw articles to feature space, we use the Stanford Natural Language Processing Group's GloVe vector embeddings [6]. These 100-dimensional vector embeddings of English words are constructed by training on co-occurrence statistics of words in a large text corpus. We create 100-dimensional embeddings of entire document fragments by

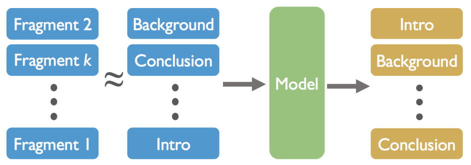

Figure 2: k-fragments approach

summing the GloVe vectors of their words, summarizing the meaning of fragments through their constituent words. One drawback of this method is that the embedding is permutation invariant, i.e., words within the fragment can be ordered arbitrarily without affecting the embedding. However, this is a computationally cheap and surprisingly effective way of capturing semantics on the fragment level, so all except one of our models utilize these features.

Another feature mapping method we use relies on n-grams, specifically unigrams and bigrams, which are the counts of words or pairs of consecutive words within a fragment. The feature space for this mapping is extremely large given the number of words and word pairs in the English language. To address this, we simply use the 5000 most-frequently used words in the corpus, excluding stop words, which we identify using Python's Natural Language Toolkit (nltk) [8]. This feature mapping method captures the relative ordering of words within a fragment and is therefore not permutation invariant. Naive Bayes did well with these features without becoming computationally intractable. Other models ran too slowly with these features.

4. Methods

We explore two variations of the document reconstruction problem. The first is the k-fragments approach, which entails reassembling a document that has been fragmented into k equally-sized fragments, which we call the k-fragment problem. For instance, letting $k = 5$, we explore the problem of computing the original permutation of 5 fragments of a source document. As evidenced in our document sample from section 4, the choice of k will roughly reflect a natural subdivision of documents into functional parts (introduction, body, conclusion), and thus the objective is to learn the features of these document structures.

The second variation, the transitional approach, entails reordering a document from fragments of m-sentences each, for fixed m (the m-sentence problem). For instance, letting $m = 1$, our problem becomes equivalent to reordering a document given a collection of its sentences, which is the traditional approach in NLP literature. The m-sentence problem inherently deals more with local flow of natural language between chunks of relatively small size.

4.1. k-Fragment Problem

This approach assumes that the contents of a document fragment depend on its position within the source document. Given this assumption, we model our problem as a maximum likelihood optimization over permutations p:

$$\max_{p} \prod_{i=1}^{k} P(pos = i \mid frag_{p_i}). \tag{1}$$

Note that pos and $frag$ represent fragment position and features, respectively.

4.2. Supervised K-means

For our baseline, we implemented a "supervised K-means" model for identifying fragment position. We use fragment GloVe vectors sums for our feature vectors. In this approach, the assignment of data points to centroids is fixed by the labels of the dataset, and we compute the k centroids $\mu_1, \dots, \mu_k$ by taking the mean of these points. To reorder a fragmented document $frag_1 || frag_2 || \cdots || frag_k$, we assign document fragments to centroids (and thus position) such that the mean squared distance between fragments and centroids is minimized. This choice of baseline and feature vector is reasonable, since we might expect fragments of the same semantic function (e.g., introduction, conclusion) to be composed of words with close GloVe vector embeddings. By capturing this clustering, we can reasonably expect this approach to outperform a random agent.

4.3. Nearest-Neighbors with kd-Tree

Going beyond this supervised K-means approach, we implement a nearest-neighbor model. In this supervised model, training points and their labels are fully stored. For a test fragment x, our algorithm identifies the $n = 15$ nearest fragments to x within the training set. Using these neighbors, we get the following learned function predicting the likelihood of x belonging in each fragment position.

$$P(pos = i|x) = \frac{1 + \#\{\text{train neighbors of } x \text{ labeled pos } i\}}{k + n}$$

Notice that the $+1$ of the numerator and $+n$ of the denominator are analogous to Laplace smoothing used for Naive Bayes; these terms guarantee that our predictor gives nonzero predictions for each label, which is critical for finding reasonable solutions to (1).

4.4. Naive Bayes

As a final approach, we expect that a standard Naive Bayes classifier should exhibit reasonable performance in estimating our function $P(pos = i \mid frag)$. The Naive Bayes assumption is essentially the assumption we have made in choosing to solve the k-fragment problem; that

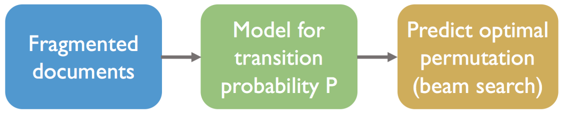
Fragmented documents
Model for transition probability P
Predict optimal permutation (beam search)

Figure 3: Transitional approach

is, given a choice of fragment position i, the distribution of words present within the fragment will be similar, despite some dependence on document context. For articles sourced from the categories "people", "films", and "settlements", the assumption is even more valid.

We implement a Naive Bayes model for the k-fragment problem. We train a Naive Bayes model for each fragment position, producing a learned function as described above. To reconstruct a document, we employ these k learned models in solving the optimization problem of (1).

4.5. m-Sentence Problem and the Transition Model

In this context, documents are scrambled into fragments of m-sentences each. Thus, documents are split into a variable number of fixed-size fragments. The transitional model assumes that the features of a fragment depend on the features of the previous fragment. I.e., it models local coherence and the fragment-to-fragment flow of information in a document. We therefore learn $P(frag_i \mid frag_{i-1})$. This leads to an optimization problem of similar nature to (1),

$$\max_{p} P(frag_{p_1}) \prod_{i=2}^{k} P(frag_{p_i} \mid frag_{p_{i-1}}). \tag{2}$$

We use our Naive Bayes model to provide $P(frag_{p_1} \mid pos = 1)$ as a prior confidence for each permutation.

In this m-sentence model, the number of fragments will often be large, and thus we require an approximation search algorithm to find a suitable ordering. We implement beam search with a beam of size 5 in order to discover a suitable permutation to maximize (2) above.

We investigated using as our baseline an approach from [2] which used the cosine similarity of fragment embeddings to model $P(frag_{p_i}|frag_{p_{i-1}})$ from (2). The intuition behind this approach is that fragments mentioning similar subjects will have high cosine similarity and will cluster after the optimization of (2). However, this approach did not perform better than a random re-ordering function, which we suspect is due to the commutativity of cosine similarity, which does not capture order of fragments.

4.6. Logistic Regression

We implemented logistic regression as a transitional model. We designed a simple logistic regression model to learn the binary classification problem "Does fragment A follow fragment B?", which is how we estimated $P(frag_i \mid frag_{i-1})$ for the m-sentence problem. Our feature vector consists of a difference of fragment GloVe vector sums; that is, if fragment X precedes fragment Y within a document, we subtract the GloVe vector embedding of Y from that of X. This choice of feature is motivated by [6], which shows that vector differences of words which co-occur in document fragments carry semantic structure.

5. Results and Discussion

We use accuracy and τ as performance metrics for our models. The accuracy of n re-ordered documents is

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{k^{(i)}}\sum_{j=1}^{k^{(i)}} 1\{\hat{y}_j^{(i)} = y_j^{(i)}\},$$

where $k^{(i)}$ is the number of fragments in the ith document, and $1\{\hat{y}_j^{(i)} = y_j^{(i)}\}$ indicates whether the jth fragment was placed in the correct position. For the same set of n re-ordered documents, the τ metric can be expressed as

$$\frac{1}{n}\sum_{i=1}^{n}\left(1 - (\text{\# pairwise inversions})^{(i)} \div \binom{k^{(i)}}{2}\right),$$

where a pairwise inversion is an instance of two fragments switching their relative order, and τ is intended to be some measure of the relative ordering compared to the correct ordering. For any document $D^{(i)}$ split into $k^{(i)}$ fragments, a random reassembly has an expected accuracy of $1/k^{(i)}$, and the expected τ value is 0.5 regardless of $k^{(i)}$. For the k-fragment problem, the accuracy metric is most relevant because our models attempt to predict the exact position of each fragment. For the m-sentence problem, the τ metric is more indicative of performance as our models in this context are designed to assess the fluency of adjacent fragments and are oblivious to fragment position.

Figure 4 shows the performance of our k-fragment approach baseline, supervised K-means, on the mixed dataset with varying k. As expected, performance drops as k grows larger because the position semantics assumption breaks down. This justifies our use of $k = 5$ as a reasonable standard for the k-fragment problem because it is of non-trivial size but is still small enough that the core assumption holds.

The nearest-neighbor model requires a parameter n indicating the number of near neighbors to select from the memorized training data. In Figure 5, we see that as we increase this parameter, the training accuracy steeply declines while test accuracy gradually increases, with only small gains for $n > 15$. We choose $n = 15$ to allow the nearest-neighbor model to generalize best to test data.

The m-sentence variant is more difficult than the k-fragment problem, as the number of fragments is generally

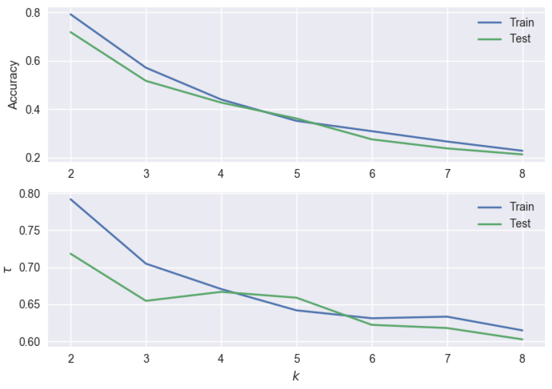

Figure 4: Supervised K-means Acc. and τ vs. k

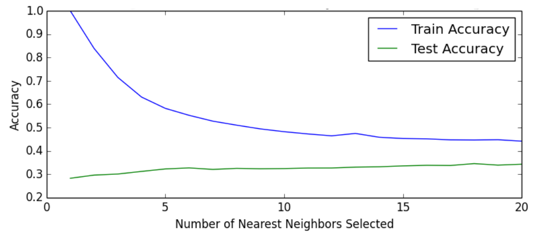

Figure 5: Train / Test Accuracy over # selected neighbors

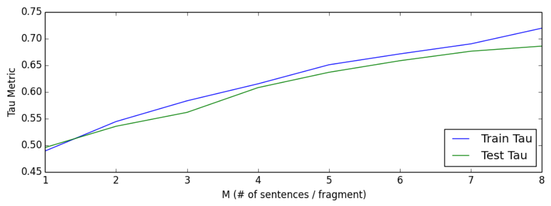

Figure 6: Performance of log. regression for choice of m

higher and the semantic content of each fragment is less. This is reflected in Figure 6, which shows that performance of our logistic regression model increases as m increases. Our choice of $m = 5$ thus represents a middle ground between k-fragments and the exceedingly difficult problem of re-ordering single-sentence fragments. With $m = 5$, the mean number of fragments encountered per dataset ranges from 6.1 for films to 8.9 for settlements.

Table 1 summarizes the performance of our models on all four categories of our Wikipedia dataset. For the k-fragments problem, the supervised K-means baseline sets a high bar with a minimum test accuracy of 0.310 on the settlements dataset and a maximum of 0.447 on the people dataset. Naive Bayes did better in some categories and about the same on others. Surprisingly, Naive Bayes did reasonably well on the settlements dataset, which K-means performed the worst on, suggesting that the increased dimensionality of the n-grams features helped elicit more se-

	Train		Test	
	τ	Acc.	τ	Acc.
k-Fragment Problem, $k = 5$				
Supervised K-means				
Mixture	.647	.352	.639	.337
People	.738	.467	.729	.447
Films	.708	.451	.701	.443
Settlements	.752	.461	.619	.310
Naive Bayes				
Mixture	.663	.330	.632	.317
People	.767	.462	.758	.445
Films	.693	.388	.692	.389
Settlements	.694	.370	.692	.389
Nearest Neighbors, $n = 15$ neighbors				
Mixture	.710	.458	.649	.340
People	.782	.562	.731	.435
Films	.828	.622	.774	.495
Settlements	.861	.697	.814	.590
m-Sentence Problem, $m = 5$				
Logistic Regression				
Mixture	.612	.318	.597	.292
People	.630	.330	.623	.314
Films	.611	.306	.620	.292
Settlements	.580	.246	.548	.200

Table 1: Computed performance metrics for each model on each dataset, for the corresponding problem.

mantics from the settlements dataset than the simple GloVe sum centroids of supervised K-means could. We also see that the nearest-neighbor algorithm achieves the highest performance on all datasets except people, where our supervised K-means model is more accurate but has a lower τ. In the m-sentence setting, our logistic regression achieves $\tau = 0.623$, which is significantly higher than a random baseline. This indicates that the regression can leverage the semantics of fragment GloVe vector differences.

As in literature [1], we tested our models on multiple datasets with distinct structures and general topics. The fact that our models perform well on all the Wikipedia categories we tested as well as an arbitrary mixture of topics shows that they can effectively capture semantic information and text structure regardless of the particular content of the documents they are used to model. All models achieved better-than-random results (accuracy $> 20\%$ and $\tau > 0.5$ for k-fragments). Unsurprisingly, the models did best on defined categories (people, film, and settlements), where constituent articles shared similar semantic structures, and did worst on mixtures of topics.

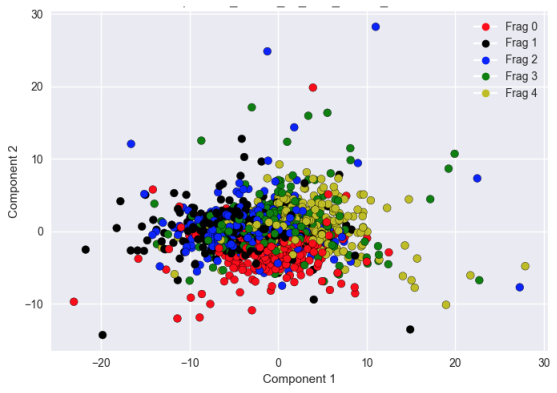

Figure 7: Principal Components Analysis of GloVe Sums

As a means to gain visual intuition for the performance of our K-means and nearest-neighbors approach, we performed principal component analysis on the GloVe sum representations of our document fragments. Figure 7 shows a plot of fragment embeddings using principal components 1 and 2; fragments are colored according to their true position in their respective documents. Note how fragments cluster according to their position, which indicates that fragments of similar position also share similar semantics.

6. Conclusions and Future Work

Problems relating to the generation of coherent of natural language text are difficult, but we made promising headway in reconstructing documents. Training and testing on articles with similar structure yielded the best results, and our very best model, KD trees for the k-fragment problem, achieved a whopping $\tau = 0.814$ on the test set of the settlements category. Our k-fragments baseline, supervised K-means, also performed decently well. This makes sense given that the fragment GloVe vector sums actually do exhibit clustering when examined using PCA, validating our assumption that position carries meaning. By comparison, the m-sentence problem is strictly harder as the number of fragments grows linearly with the size of the documents. Nonetheless, we were able to achieve better-than-random accuracy and τ metrics in reordering texts with the transitional model driven by logistic regression.

In future work, we would like to relax our constraints on fragment size and number, which are already more flexible than those in literature, to explore a more general document-reconstruction problem. To do so, we might improve upon the transitional model by using recurrent neural networks to model $P(frag_{p_i} \mid frag_{p_{i-1}} \ldots frag_{p_1})$ instead of just $P(frag_{p_i} \mid frag_{p_{i-1}})$ [1]. We would also like to move past simple GloVe sums and instead use unsupervised representation learning to generate fragment embeddings.

7. Contributions

Both teammates put significant work into this project. Mitchell worked on nearest-neighbors, logistic regression, and naive Bayes. Caelin worked on supervised and unsupervised k-means, principal components analysis, and cosine similarity. Both contributed to the pipeline/test harness for models, experimental design, and Wikipedia data preparation. Both worked on the milestone, poster, and final report.

References

[1] L. Logeswaran, H. Lee, & D. Radev. *Sentence Ordering using Recurrent Neural Networks.* arXiv preprint (arXiv:1611.02654 [cs.CL]).

[2] P. W. Foltz, W. Kintsch, and T. K. Landauer. The measurement of textual coherence with latent semantic analysis. Discourse processes, 25(2-3):285307, 1998.

[3] M. Lapata. Probabilistic text structuring: Experiments with sentence ordering. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1, pages 545552. Association for Computational Linguistics, 2003.

[4] X. Chen, X. Qiu, and X. Huang. Neural sentence ordering. arXiv preprint arXiv:1607.06952, 2016.

[5] R. Barzilay and M. Lapata. Modeling local coherence: An entity-based approach. Computational Linguistics, 34(1):134, 2008.

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[7] L. White, R. Togneri, W. Liu and M. Bennamoun, ”Modelling Sentence Generation from Sum of Word Embedding Vectors as a Mixed Integer Programming Problem,” 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, 2016, pp. 770-777. doi: 10.1109/ICDMW.2016.0113

[8] Natural Language Toolkit. (http://www.nltk.org/)

[9] WikiMedia Dumps. English Wikipedia Dataset. (meta.wikimedia.org/wiki/Data_dump_torrents)

[10] WikiExtractor. (https://github.com/attardi/wikiextractor)