WikiSeq: Mining Maximally Informative Simple Sequences from Wikipedia

Goutam Nair, Vikram Pudi
Centre for Data Engineering
IIIT Hyderabad, India
goutam.nair@students.iiit.ac.in, vikram@iiit.ac.in

Abstract

The problem of ordering documents in a large collection into a sequence that is efficient for learning (both human and machine) is of high practical significance, but has not yet been well-formulated. We formulate this problem as mining a maximally informative simple sequence of documents. The mined sequence should be maximally informative in the sense that the reader learns quickly by reading only a few documents, and it should be simple so that the reader is not overwhelmed while trying to learn the content. The task can be posed as: Given that a reader wishes to read (at most) $k$ documents, which documents should be selected from the repository and in what order, so as to provide maximum information. We present the WikiSeq algorithm for this purpose. We also design a metric based on information-gain to help objectively evaluate WikiSeq, and conduct experiments to compare with indicative baselines. Finally, we provide case-studies to subjectively illustrate WikiSeq’s merits.

1 Introduction

Large collections of documents that link or cite each other exist on the internet, such as in Wikipedia, and also in research paper repositories. Their rich citation structure allows readers to browse documents in their own way, and they serve as excellent reference material. Yet a common and critical requirement is to provide a default ordering of the documents, especially to readers who are reading for the first time for the purpose of learning, and not for reference.

In the absence of a default sequence, learning about entirely new topics can be exhausting, especially if the topic provides copious amounts of information and delves deep into the subject matter. Facing vast amounts of information and not knowing where to start and what to read next can turn potential learning experiences into episodes of frustration and confusion. Producing these sequences manually is labor-intensive and requires a deep knowledge of the topic.

In this paper, we consider the task of automatically ordering textual documents in a collection into a sequence that is efficient for learning. We assume that the documents in the collection link or cite each other, such as in Wikipedia, or in research paper repositories. Although practically significant, this problem has not yet been well formulated. Examples of such orderings include syllabus sequences in textbooks where a sequence of sub-topics is followed for learning a topic.

Wikipedia, the world’s largest encyclopedia, provides an enormous collection of documents that is carefully curated and categorized. These categories range from Philosophy and Food culture to Nanotechnology and Aeronautics. Anybody wishing to expand their horizons on any of these topics would face an ocean of information, which has been divided into subcategories and pages. Many of these subcategories are not entirely relevant to the main topic or contain complex information that is not pertinent to the main topic.

Figure 1: CategoryTree for the topic Arithmetic

Figure 1 shows an example of a CategoryTree for the category Arithmetic. As seen, some of the subcategories are either irrelevant (Black Holes) or too complicated (Differential Calculus, Spatial Gradient, etc.) for anyone wishing to learn Arithmetic.

Topic learning sequences can benefit machine learning approaches like word2vec (Mikolov et al. 2013), paragraph2vec (Le and Mikolov 2014) and deep learning based techniques that require huge amounts of data. Instead of applying these algorithms on the entire corpus, it is worth exploring (in future) if they may be equally or more effective by applying them on the first $k$ documents of a good topic learning sequence.
To address the problem of ordering documents in Wikipedia, we investigate several techniques that provide us with topic sequences by utilizing the use of topic names and inter-document relationships in the corpus. We then devise an algorithm to select the next document of the sequence based on its information gain, ubiquity and other features.

The contributions of our paper are as follows:

1. We formulate the problem of ordering documents in large collections into a sequence that is efficient for learning.
2. We propose WikiSeq, a simple but effective solution for this problem.
3. We provide a metric based on information gain for objectively evaluating various ordering approaches.
4. We show experimentally that WikiSeq results in a faster word2vec learning speed than other techniques. Finally, we use Wikipedia categories that are easy to interpret so as to subjectively illustrate the merits of WikiSeq.

The rest of the paper is organized as follows: Section 2 gives a brief overview on related research work. Section 3 formalizes the problem statement and Section 4 describes our approach to the problem in detail including the intuition and mechanics of the algorithm. Sections 5 and 6 present the experimental results and conclude the work, respectively.

2 Related Work

The task of ordering textual content has been studied extensively, focusing on ordering information for single and multi-document summarization and sentence ordering (Barzilay, Elhadad, and McKeown 2002; Radev and McKeown 1998; Lapata 2003; Elhadad and McKeown 2001). A departure in our work is that we devise an algorithm to output an ordering of documents from a corpus of documents. Also, we use a novel metric to present and compare the results and show that the topic sequences obtained from our approach produces the best results.

Furthermore, there has been much research done on structuring textual content in a corpus (Barzilay 2010; Chen, Snyder, and Barzilay 2007; Chen et al. 2009). A related field of study is structuring content on Wikis by using relevant properties of Wiki systems (Reinhold 2006; Haake, Lukosch, and Schümer 2005). These approaches employ a means of navigating the paths in Wikis that are useful in bringing a structure to textual content. Such approaches have been effective in identifying relationships between articles. In this work, we incorporate such relationships between Wikipedia documents into a corpus-based approach to ordering.

Vector representation of documents through probabilistic models (Blei, Ng, and Jordan 2003) and distributed representations of words and phrases (Mikolov et al. 2013; Le and Mikolov 2014) has received much recent attention. While word2vec and paragraph2vec learn representations for words and paragraphs based on semantic similarity, our task requires a probabilistic distribution of topics across the corpus. We propose a technique to represent document vectors based on probability and compare the results of our model with those obtained using Latent Dirichlet Allocation.

The rise of eLearning systems and usage of Wikipedia for didactic purposes (Reinhold and Abawi 2006; Limongelli, Gasparetti, and Sciarro 2015) necessitates a sequencing algorithm to create viable learning strategies for a topic. Our algorithm aims to create these learning sequences through unsupervised learning methods.

3 Problem Formulation

The problem is to order a given collection of documents \( C \), where each document \( d \) is represented as \( (T, V) \). \( T(d) \) represents the title of the document \( d \) and \( V(d) \) represents the document as a probabilistic distribution of topics.

\[
V(d_i) = (p_{i1}, p_{i2}, p_{i3}, \ldots, p_{iN})
\]

We consider the usage of document title and topic to be equivalent, in this paper. This works fine in the context of Wikipedia articles, as topics in a page are links to other pages that describe the topic in detail; and the anchor text of those links is the title of the cited page.

Intuitively, the sequence of documents should be maximally informative and simple. Being maximally informative captures the idea that the reader learns quickly by reading only a few documents, and being simple means that the reader is not overwhelmed while learning the content. We formalize these two concepts in the following sub-sections.

A more general task than above can be posed as: Given that a reader already knows the content of some specified documents in \( C \) and wishes to read (at most) \( k \) more documents from \( C \), which documents should be selected and in what order, so as to provide maximum information?

Any greedy constructive algorithm that solves the simpler ordering problem posed first must be capable of selecting the next document that should appear at each stage after having selected the previous sub-sequence of documents. Hence, it should be possible to extend the algorithm to handle the case when the user has marked some documents as known or already read. Second, it should be possible to stop the algorithm after having generated a sequence of \( k \) documents.

In any case, in this paper, we explore only such greedy constructive algorithms, and hence they can easily handle the more general task posed above.

Maximum Information Gain

We define two measures to help formalize the notion of a maximally informative sequence. The Knowledge Measure for a sequence \( S_n \) is given as:

\[
Knowledge \ Measure = (K_1, K_2, K_3, \ldots, K_N)
\]

where,

\[
K_j = \frac{\sum_{d_i \in S_n} p_{ij}}{\sum_{d_i \in C} p_{ij}}
\]

and \( N \) is the number of topics in collection \( C \). Note that \( S_n \) represents the current state of a growing sequence.

The Knowledge Measure helps us capture the amount of information content in the mind of the reader after reading the documents in a sequence \( S_n \). It intuitively corresponds to the “strength” of each topic in \( S_n \).
The Knowledge Rate for a sequence $S_n$ can be represented as:

$$\text{Knowledge Rate} = \text{Entropy}(\frac{K_1}{|S_n|}, \frac{K_2}{|S_n|}, \frac{K_3}{|S_n|}, \ldots, \frac{K_N}{|S_n|})$$

(4)

The Knowledge Rate roughly corresponds to the normalized information content (entropy) in $S_n$, which helps us gauge the rate of information gain for a sequence. This measure allows us to compare the efficacy of the topic sequence. The Knowledge Rate at step $n$ is defined as $S_{n-1}$ changes to $S_n$.

### Simplicity Problem

A vital aspect of creating learning strategies is sequences is that documents that occur early in the sequence must be complicated or require other prior knowledge. For instance, consider the task of ordering documents in the category Data structures. In this category, Binary Search Trees is present in a large number of documents and renders a large information gain because of its necessity in many applications. However, a person wishing to learn about data structures will learn some of the simpler structures like Arrays or Linked lists before moving on to more complex data structures.

We propose techniques to tackle this problem that provide a trade-off between documents with higher information gain and simpler documents to create sequences that are efficient learning strategies.

### 4 Our Approach

In this section, we introduce our approach to ordering documents. First, we present a naive approach to topic sequencing and then devise gSeq, a more sophisticated algorithm that builds on its concepts. We finally culminate in the WikiSeq algorithm whose workflow diagram is shown in Figure 2.

#### Greedy-LDA Algorithm

For this algorithm, we represent each document $d$ as a mixture of latent theme probabilities $V(d)$ obtained using Latent Dirichlet Allocation, LDA (Blei, Ng, and Jordan 2003). Since our task needs topic probabilities instead of context-inferred similarity scores, we do not consider the usage of more recent approaches like paragraph2vec. Using LDA in conjunction with an approach to estimate the natural number of topics in the corpus (Arun et al. 2010), we compute $V(d)$ but with very large overheads in execution time. Note that WikiSeq itself does not use LDA to define $V(d)$.

The sequence is then determined greedily at each step by selecting the document with highest information gain. The information gain provided by a document is calculated as the Kullback-Leibler divergence from the Knowledge Measure of the current sequence. A larger divergence implies a greater information gain from the document. It is calculated as:

$$IG(d_i) = \sum_{j=1}^{n} p_{ij} \cdot \log \frac{p_{ij}}{q_j}, \quad p_{ij} \in V(d_i)$$

(5)

This method does not factor in the relationships between documents nor does it tackle the simplicity problem. It will be shown that a sequence determination of this kind falls substantially behind other approaches in the experimental sections.

#### The gSeq Algorithm

In this graph-based sequencing (gSeq) algorithm, we represent documents as probabilistic distributions of document titles and category names. It has been shown in (Schönhofen 2009) that category names are effective in identifying topics discussed within the content of the document in Wikipedia.

More precisely, we propose that a document $d$ be represented as a probabilistic distribution $V(d)$ of other document titles $T_j$ that appear in its content.

$$p_{ij} = \frac{\text{Count of } T_j \text{ in } d_i}{\sum_{T \in V} \text{Count of } T \text{ in } d_i}$$

(6)

Now, we construct an undirected graph to capture the relationship between documents within a category. This is illustrated in Figure 3, where $T_i$ are document titles and $W_{ij}$ is the edge weight between nodes, $T_i$ and $T_j$.

#### Document Set

**Document Set:** We define the document set for a title $T$, $\delta(T)$ as the set of documents that contain $T$ in their content.
Then, the edge weight for any pair of nodes, $T_i$ and $T_j$ is:

$$W_{ij} = |δ(T_i) \cap δ(T_j)|$$  \hspace{1cm} (7)

This edge weight represents the co-occurrence of titles in the corpus. A greater co-occurrence of titles implies a stronger relationship between documents. We use this to prune out documents that are not closely related to the main topic. This graph structure allows us to leverage the power of inter-document relationships when creating the sequence.

In $gSeq$, the next document in the sequence is chosen by its benefit value. The benefit value incorporates the information gain from Greedy-LDA (Equation 5) and also integrates the number of related documents (edges, $E(d)$) obtained from the graph structure. The benefit value for document $d_i$ is given as:

$$B(d_i) = α \cdot IG(d_i) + β \cdot |E(d_i)|$$  \hspace{1cm} (8)

To tackle the simplicity problem and place simpler documents earlier in the sequence, we devise the topic set approach.

**Topic Set:** The topic set, $τ(d)$ for a document $d$, is defined as the set of document titles $T_i$ that are present in the content of the document $d$.

Here, we consider a significant portion of the document’s content to be comprised of content from other documents. Now, we consider a document $d'$ to be simpler if

$$τ(d') \subset τ(d)$$  \hspace{1cm} (9)

That is, $d'$ is simpler as it has fewer topics than $d$.

**WikiSeq**

The WikiSeq algorithm builds on $gSeq$. While it retains the document representation and graph structure to represent relationships, the algorithm improves the sequence determination by considering the ubiquity score of the document title. The ubiquity score captures the pervasiveness of the document title across the collection. It is calculated as:

$$μ(T) = \log \frac{|δ(T)|}{N}$$  \hspace{1cm} (10)

$\delta(T)$ is the document set for the title $T$ and $N$ is the total number of documents in the collection.

The ubiquity score helps better determine more general and broader concepts, thereby improving the learning sequence. Thus, the benefit value to determine the next document is improved to be:

$$B(d_i) = α \cdot IG(d_i) + β \cdot |E(d_i)| + γ \cdot μ(T(d_i))$$  \hspace{1cm} (11)

We empirically determine the values for $α = 0.2, β = 0.3$ and $γ = 0.5$ based on our experiments.

Also, we develop an alternate strategy to tackle the simplicity problem by considering ubiquity score of the document title. We consider a document $d'$ to be simpler, if it has a higher ubiquity score than document $d$, and

$$T(d') \in τ(d)$$  \hspace{1cm} (12)

Our reasoning is that a topic that is part of $d$ and more pervasive across the corpus should be learnt as a prerequisite.

Also, this approach transcends the topic set approach because it does not require all topics in $d'$ to be a subset of $d$, since documents can often include topics that are reasonably complex in order to express ideas. For instance, the document ‘Arrays’ may include ‘Hash Tables’, which is a more complex topic, as an application. Due to this, we find that the topic set strategy misses out many simplicity relationships between documents.

The pseudo code of our approach is presented in Algorithm 1. In lines 5-13, we create an inverted index to compute the document set for each title in the collection. The process of generating the probabilistic distribution for each document is show in lines 14-17. The edges for each graph topic is enumerated in lines 18-21. Then, we prune the graph structure in line 23 by removing edges with low weights and isolated nodes using $PruneGraph$ and determine the next document of the sequence in line 25. Finally, we tackle the simplicity problem by leveraging the ubiquity score of the document title in line 26.

**Algorithm 1 WikiSeq Algorithm**

1: **Input:** Document Collection: $C$
2: **Output:** Learning Sequence: $S$
3: $T \leftarrow$ Titles of documents $D$
4: $Sequence \leftarrow \emptyset$
5: for all $t \in T$ do
6:   $δ_t \leftarrow \emptyset$
7:   for all $d \in C$ do
8:      $count_{t,d} \leftarrow \text{Count of } t \text{ in } d.content$
9:      if $t \in d.content$ then
10:         $δ_t \leftarrow δ_t \cup \{d.title\}$
11:     end if
12:   end for
13: end for
14: for all $t \in T$ do
15:   $sum_t \leftarrow \sum count_{t,d} \forall d \in C$
16:   $V(d)_t \leftarrow [(\text{Count of } t \text{ in } d)/sum_t] \forall d \in C$
17: end for
18: $E \leftarrow \emptyset$
19: for all $(t_1, t_2) \in (T \times T)$ do
20:   $E \leftarrow E \cup \{δ_{t_1} \cap δ_{t_2}\}$
21: end for
22: $G = CreateGraph(T, E)$
23: $G' = PruneGraph(G)$
24: for all nodes $n \in G'$ do
25:   $X \leftarrow \text{Title with max. } B(d) \forall d \in G'$
26:   $T_{next} \leftarrow \text{Document title with max. } μ(T') \forall T' \in τ(X)$
27: end for
28: **Sequence.append($T_{next}$)**

5 **Experiments and Results**

**Data Preparation**

In our experiments, our corpus consisted of Wikipedia articles from different categories including Data structures, Algorithms and Data structures, Arithmetic, etc. Each category and resulting subcategories have designated articles that contain information about the topic. We use this article as the content for the topic and construct the vector $V(d)$. Algorithms and Data structures contain an excess of 500
**Evaluation Measures**

We evaluate our model using *Knowledge Rate* (Equation 4) to judge the efficiency of our results. Since, the number of topics in the sequence can be different depending on the algorithm used (graph-based algorithms prune some topics), comparing the rate of knowledge increase justifies the fairness of our evaluation. We also report and compare the topic sequences obtained to qualitatively assess the results.

**Baselines**

Our first three baselines correspond to naive sequencing algorithms. Regular Breadth-First Search (BFS) and Depth-First Search (DFS) utilize Wikipedia’s categorical hierarchy structure to sequence documents. The Random ordering method selects a random ordering of the topics.

We also compare WikiSeq against two competitive baselines: Greedy-LDA and gSeq. Greedy-LDA selects topic sequences solely based on maximum information gain, without regard to the *simplicity problem* as discussed in Section 3. The gSeq algorithm employs an approach different from WikiSeq to tackle the simplicity problem and uses an unrefined heuristic to determine the sequence.

**Results**

Table 1 presents and compares the first 20 documents titles of the ordering produced by WikiSeq when applied to the category *Algorithms and Data structures*. We compare WikiSeq against a competitive baseline and a naive baseline that uses Wikipedia’s category tree structure.

```
<table>
<thead>
<tr>
<th>WikiSeq</th>
<th>Greedy-LDA</th>
<th>BFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>Algorithms</td>
<td>Algorithms</td>
</tr>
<tr>
<td>Arrays</td>
<td>Abstract data types</td>
<td>Abstract data types</td>
</tr>
<tr>
<td>Data structures</td>
<td>Algorithmic complexity attacks</td>
<td>Algorithmic complexity attacks</td>
</tr>
<tr>
<td>Polynomials</td>
<td>Analysis of algorithms</td>
<td>Analysis of algorithms</td>
</tr>
<tr>
<td>Linear programming</td>
<td>Analysis of parallel algorithms</td>
<td>Data structures</td>
</tr>
<tr>
<td>Search algorithms</td>
<td>Checksum algorithms</td>
<td>Priority queues</td>
</tr>
<tr>
<td>Trees (data structures)</td>
<td>Calendar algorithms</td>
<td>Computer algebra</td>
</tr>
<tr>
<td>Genetic algorithms</td>
<td>Combinatorial algorithms</td>
<td>Algorithm description languages</td>
</tr>
<tr>
<td>Signal processing</td>
<td>Cyclic redundancy checks</td>
<td>Algorithmic trading</td>
</tr>
<tr>
<td>Dynamic programming</td>
<td>International Standard Book Number</td>
<td>Approximation algorithms</td>
</tr>
<tr>
<td>Recursion</td>
<td>Concurrent algorithms</td>
<td>Bioinformatics algorithms</td>
</tr>
<tr>
<td>Numerical analysis</td>
<td>Data mining algorithms</td>
<td>Calendar algorithms</td>
</tr>
<tr>
<td>Compression algorithms</td>
<td>Classification algorithms</td>
<td>Checksum algorithms</td>
</tr>
<tr>
<td>Search trees</td>
<td>Data clustering algorithms</td>
<td>Combinatorial algorithms</td>
</tr>
<tr>
<td>Analysis of algorithms</td>
<td>External memory algorithms</td>
<td>Compression algorithms</td>
</tr>
<tr>
<td>Hashing</td>
<td>Computational number theory</td>
<td>Computer arithmetic algorithms</td>
</tr>
<tr>
<td>Monte Carlo methods</td>
<td>Pseudorandom number generators</td>
<td>Concurrent algorithms</td>
</tr>
<tr>
<td>Randomized algorithms</td>
<td>Stochastic algorithms</td>
<td>Cryptographic algorithms</td>
</tr>
<tr>
<td>Linked lists</td>
<td>Concurrency control algorithms</td>
<td>Data mining algorithms</td>
</tr>
<tr>
<td>Heuristic algorithms</td>
<td>Combinatorial optimization</td>
<td>Database algorithms</td>
</tr>
</tbody>
</table>
```

Table 1: Comparison of topic orderings for *Algorithms and Data structures*

We notice that the sequence from WikiSeq very closely represents a viable learning sequence for anyone wanting to learn about the category. In contrast, the other topic orderings include very complex topics (*Checksum algorithms*, *Bioinformatics algorithms*) that involve other prior knowledge that is not already in the sequence. This highlights the capability of WikiSeq in handling the *simplicity problem*.

Furthermore, Figure 4 which shows the *Knowledge Rate* indicates that the simple baseline algorithms (BFS, DFS, Random) fall behind the more sophisticated graph-based approaches (WikiSeq, gSeq). The results show that WikiSeq performs substantially better than the other algorithms.

This considerable benefit occurs because of representing documents as document-title probabilities instead of latent themes in the corpus and refining the estimation to include *ubiquity scores*. The downward tick in the rate of information gain occurs because we include simpler documents earlier in the sequence, albeit with lower information gain.
To analyze the performance on smaller categories, we winnow down the previous dataset and observe our results for Data structures, a subcategory of Algorithms and Data structures. Notice in Table 2 that the output of WikiSeq resembles one that could be manually ordered based on complexity and prerequisites, demonstrating its efficacy.

Figure 5 shows the knowledge rate for the two graph-based approaches. We see that the usage of ubiquity score and a difference in tackling the simplicity problem causes WikiSeq to perform better.

We also investigate the performance of WikiSeq in the category Arithmetic against Greedy-LDA, which uses Latent Dirichlet Allocation to represent documents. Table 3 presents and compares this ordering of the first 20 document titles in the sequence. Figure 6 depicts a comparison of WikiSeq against this baseline, which clearly demonstrates the effectiveness of capturing inter-document relationships and ubiquity scores.

It should be noted that WikiSeq clearly outperforms the other approaches in providing qualitatively superior learning sequences and a faster rate of knowledge gain.

### Table 2: Comparison of topic orderings for Data structures

<table>
<thead>
<tr>
<th>WikiSeq</th>
<th>gSeq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data structures</td>
<td>Data structures</td>
</tr>
<tr>
<td>Arrays</td>
<td>Trees (graph theory)</td>
</tr>
<tr>
<td>Trees (data structures)</td>
<td>Binary trees</td>
</tr>
<tr>
<td>Search trees</td>
<td>Arrays</td>
</tr>
<tr>
<td>Linked lists</td>
<td>Distributed data structures</td>
</tr>
<tr>
<td>Binary trees</td>
<td>Spanning tree</td>
</tr>
<tr>
<td>Heaps (data structures)</td>
<td>B-tree</td>
</tr>
<tr>
<td>B-tree</td>
<td>Heaps (data structures)</td>
</tr>
<tr>
<td>Priority queues</td>
<td>Search trees</td>
</tr>
<tr>
<td>Trees (graph theory)</td>
<td>Priority queues</td>
</tr>
<tr>
<td>Succinct data structure</td>
<td>Linked lists</td>
</tr>
<tr>
<td>Trees (set theory)</td>
<td>Trees (set theory)</td>
</tr>
<tr>
<td>Associative arrays</td>
<td>Succinct data structure</td>
</tr>
<tr>
<td>Spanning tree</td>
<td>K-tree</td>
</tr>
<tr>
<td>Distributed data structures</td>
<td>Functional data structures</td>
</tr>
<tr>
<td>Probabilistic data structures</td>
<td>Trees (data structures)</td>
</tr>
<tr>
<td>K-tree</td>
<td>Hash based data structures</td>
</tr>
<tr>
<td>Kinetic data structures</td>
<td>Graph data structures</td>
</tr>
</tbody>
</table>

### Table 3: Comparison of topic orderings for Arithmetic

<table>
<thead>
<tr>
<th>WikiSeq</th>
<th>Greedy-LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic</td>
<td>Arithmetic</td>
</tr>
<tr>
<td>Ratios</td>
<td>Computer arithmetic</td>
</tr>
<tr>
<td>Rates</td>
<td>Division (mathematics)</td>
</tr>
<tr>
<td>Pi</td>
<td>Ratios</td>
</tr>
<tr>
<td>Means</td>
<td>Dimensionless ratios</td>
</tr>
<tr>
<td>Addition</td>
<td>Picture aspect ratios</td>
</tr>
<tr>
<td>Integers</td>
<td>Computer arithmetic algorithms</td>
</tr>
<tr>
<td>Multiplication</td>
<td>Addition</td>
</tr>
<tr>
<td>Fractions (mathematics)</td>
<td>Formal theories of arithmetic</td>
</tr>
<tr>
<td>Division (mathematics)</td>
<td>Subtraction</td>
</tr>
<tr>
<td>Rational numbers</td>
<td>Adders (electronics)</td>
</tr>
<tr>
<td>Percentages</td>
<td>Percentages</td>
</tr>
<tr>
<td>Rational functions</td>
<td>Data unit</td>
</tr>
<tr>
<td>Subtraction</td>
<td>Statistical ratios</td>
</tr>
<tr>
<td>Partial fractions</td>
<td>Shift-and-add algorithms</td>
</tr>
<tr>
<td>Egyptian fractions</td>
<td>Engineering ratios</td>
</tr>
<tr>
<td>Geometric series</td>
<td>Binary prefixes</td>
</tr>
<tr>
<td>Fibonacci numbers</td>
<td>Financial ratios</td>
</tr>
<tr>
<td>Golden ratio</td>
<td>Geometric series</td>
</tr>
<tr>
<td>Density</td>
<td>IEC prefixes</td>
</tr>
</tbody>
</table>

### Figure 5: WikiSeq vs gSeq for Data structures

### Figure 6: WikiSeq vs Greedy-LDA for Arithmetic

### 6 Conclusion and Future Work

We introduced and formalized the concept of topic ordering in large collections of documents and presented a novel corpus-based approach for this task. The main contribution of our work is the incorporation of inter-document relationships to create topic sequences for efficient information gain. Our approach to order the documents also takes into consideration the complexity of documents. We also define a quantitative basis for the evaluation of our results against other baselines.

While WikiSeq is designed to aid human learning, it is also likely to have applications in machine learning. We conjecture that approaches like word2vec, paragraph2vec and deep learning based text analysis techniques that require huge amounts of data, could be equally or more effective by applying them on the first $k$ documents output by WikiSeq, instead of on the entire corpus. This possibility is not further explored in this paper, and is left as future work.
References


