Hierarchical Topical Segmentation with Affinity Propagation

Anna Kazantseva
School of Electrical Engineering
and Computer Science
University of Ottawa
Ottawa, Ontario, Canada
`ankazant@eecs.uottawa.ca`

Stan Szpakowicz
School of Electrical Engineering
and Computer Science
University of Ottawa
Ottawa, Ontario, Canada
`szpak@eecs.uottawa.ca`

Abstract

We present a hierarchical topical segmenter for free text. Hierarchical Affinity Propagation for Segmentation (*HAPS*) is derived from a clustering algorithm Affinity Propagation. Given a document, *HAPS* builds a topical tree. The nodes at the top level correspond to the most prominent shifts of topic in the document. Nodes at lower levels correspond to finer topical fluctuations. For each segment in the tree, *HAPS* identifies a segment centre – a sentence or a paragraph which best describes its contents. We evaluate the segmenter on a subset of a novel manually segmented by several annotators, and on a dataset of Wikipedia articles. The results suggest that hierarchical segmentations produced by *HAPS* are better than those obtained by iteratively running several one-level segmenters. An additional advantage of *HAPS* is that it does not require the "gold standard" number of segments in advance.

1 Introduction

When an NLP application works with a document, it may benefit from knowing something about this document's high-level structure. Text summarization (Haghighi and Vanderwende, 2009), question answering (Oh et al., 2007) and information retrieval (Ponte and Croft, 1998) are some of the examples of such applications. Topical segmentation is a lightweight form of such structural analysis: given a sequence of sentences or paragraphs, split it into a sequence of *topical segments*, each characterized by a certain degree of topical unity. This is particularly useful for texts with little structure imposed by the author, such as speech transcripts, meeting notes or literature.

The past decade has witnessed significant progress in the area of text segmentation. A number of topical segmenters have been implemented (Malioutov and Barzilay, 2006; Eisenstein and Barzilay, 2008; Kazantseva and Szpakowicz, 2011; Misra et al., 2011; Du et al., 2013). Most of them can only produce single-level segmentation, a worthy endeavour in and of itself. Yet, to view the structure of a document linearly, as a sequence of segments, is in certain discord with most theories of discourse structure, where it is more customary to consider documents as trees (Mann and Thompson, 1988; Marcu, 2000; Hernault et al., 2010; Feng and Hirst, 2012) or graphs (Wolf and Gibson, 2006). Regardless of the theory, we hypothesize that it may be useful to have an idea about fluctuations of topic in documents beyond the coarsest level. It is the contribution of this work that we develop, implement and do our best to evaluate such a hierarchical segmenter.

The segmenter described here is *HAPS* – Hierarchical Affinity Propagation for Segmentation. It is closely based on a graphical model for hierarchical clustering called *Hierarchical Affinity Propagation* (Givoni et al., 2011). It is a similarity-based segmenter. It takes as input a matrix of similarities between atomic units of text in the sequence to be segmented (sentences or paragraphs), the desired number of levels in the topical tree and, for each datapoint and each level, a preference value. This value captures *a priori* belief about how likely it is that this datapoint is a segment centre at that level. The preference values also control the granularity of segmentation: how many segments are to be identified at each level. The output is a topical tree. For each segment at every level, *HAPS* also finds a segment centre (*SC*), a datapoint which best describes the segment.

The objective function maximized by the segmenter is net similarity – the sum of similarities between all segment centres and their children for all levels of the tree. This function is similar to the objective function of the well-known *k-means* algorithm, except that here it is computed hierarchically.

It is not easy to evaluate *HAPS*. We are not aware of comparable hierarchical segmenters other than that in (Eisenstein, 2009) which, unfortunately, is no longer publicly available. Therefore we compared the trees built by *HAPS* to the results of running iteratively two state-of-the-art flat segmenters. The results are compared on two datasets. A set of Wikipedia articles was automatically compiled by Carroll (2010). The other set, created to evaluate *HAPS*, consists of nine chapters from the novel *Moonstone* by Wilkie Collins. Each chapter was annotated for hierarchical structure by 3-6 people.

The evaluation is based on two metrics, *windowDiff* (Pevzner and Hearst, 2002) and *evalHDS* (Carroll, 2010). Both metrics are less then ideal and do not give a complete picture of the quality of topical segmentations. The preliminary results, however, suggest that running a global model for hierarchical segmentation produces better results then iteratively running flat segmenters. Compared to the baseline segmenters, *HAPS* has an important practical advantage. It does not require the number of segments as an input; this requirement is customary for most flat segmenters.

We also made a rough attempt to evaluate the quality of the *SC*s identified by *HAPS*. Using 20 chapters from several novels of Jane Austen, we compared the *SC*s identified for each chapter against summaries produces by a recent automatic summarizer *CohSum* (Smith et al., 2012). The basis of comparison was the ROUGE metric (Lin, 2004). While far from conclusive, the results suggest that *SC*s identified by *HAPS* are rather comparable with the summaries produced by an automatic summarizer.

A Java implementation of *HAPS* and the corpus of hierarchical segmentations for nine chapters of *Moonstone* will be made publicly available. We consider these to be the main contributions of this research.

2 Related Work

Most work on topical text segmentation has been done for single-level segmentation. Contemporary approaches usually rely on the idea that topic shifts can be identified by finding shifts in the vocabulary (Youmans, 1991). We can distinguish between local and global models for text segmentation. Local algorithms for topical segmentation have a limited view of the document to be segmented. For example, TextTiling (Hearst, 1997) operates by sliding a window through the input sequence and computing similarity between adjacent units. By identifying "valleys" in similarities, TextTiling identifies topic shifts. More recently, Marathe (2010) used lexical chains and Blei and Moreno (2001) used Hidden Markov Models. Such methods are usually very fast, but can be thrown off by small digressions in the text.

Among global algorithms, we can distinguish generative probabilistic models and similarity-based models. Eisenstein and Barzilay (2008) model a document as sequence of segments generated by latent topic variables. Misra et al. (2011) and Du et al. (2013) have similar models. Malioutov and Barzilay (2006) and (Kazantseva and Szpakowicz, 2011) use similarity-based representations. Both algorithms take as input a matrix of similarities between sentences of the input document; the former uses graph cuts to find cohesive segments, while the latter modifies a clustering algorithm to perform segmentation.

Research on hierarchical segmentation has been more scarce. Yaari (1997) produced hierarchical segmentation by agglomerative clustering. Eisenstein (2009) used a Bayesian model to create topical trees, but the system is regrettably no longer publicly available. Song et al. (2011) develop an algorithm for hierarchical segmentation which iteratively splits a document in two at a place where cohesion links are weakest. A second pass transforms a deep binary tree into a shallow and broad structure.

Any flat segmenter can certainly be used iteratively to create trees of segments by subdividing each segment, but this may be problematic. Topical segmenters are not perfect, so running them iteratively is likely to compound the error. Most segmenters also require the number of segments as an input. This estimate is feasible for flat segmentation. To know in advance the number of segments and sub-segments at each level is not a realistic requirement when building a tree.

This work describes a hierarchical model of text segmentation. It takes a global view of the document and of the topical hierarchy. Each iteration attempts to find the best assignment of segments for the whole tree. It does not need to know the exact number of segments. Instead, it takes a more abstract parameter, preference values, to specify the granularity of segmentation at each level. For each segment it also outputs a segment centre (*SC*), a unit of text which best captures the contents of the segment.

3 Creating a Corpus of Hierarchical Segmentations

Before embarking on the task of building a hierarchical segmenter, we wanted to study how people perform such a task. We also needed a benchmark corpus which could be used to evaluate the quality of segmentations produced by *HAPS*.

To this end, we annotated nine chapters of the novel *Moonstone* for hierarchical structure. We settled on these data because it is a subset of a publicly available dataset for flat segmentation (Kazantseva and Szpakowicz, 2012). In our study, each chapter was annotated by 3-6 people (4.8 on average). The annotators, undergraduate students of English, were paid $50 dollars each.

Procedure. The instructions asked the annotator to read the chapter and split it into top-level segments according to where there is a perceptible shift of topic. She had to provide a one-sentence description of what the segment is about. The procedure had to be repeated for each segment all the way down to the level of individual paragraphs. Effectively, the annotators were building a detailed hierarchical outline for each chapter.

Metrics. Two different metrics helped estimate the quality of our hierarchical dataset: *windowDiff* (Pevzner and Hearst, 2002) and *S* (Fournier and Inkpen, 2012).

windowDiff is computed by sliding a window across the input sequence and checking, for each window position, if the number of reference breaks is the same as the number of breaks in the hypothetical segmentation. The number of erroneous windows is then normalized by the total number of windows. In the Equation 1 N is the length of the input sequence, k is the size of the sliding window.

$$winDiff = \frac{1}{N-k}\sum_{i=1}^{N-k}(|ref - -hyp| \neq 0) \tag{1}$$

windowDiff is designed to compare sequences of segments, not trees. That is why we compute *windowDiff* for each level between each pair of annotators who worked on the same chapter. It should be noted that *windowDiff*is a penalty metric: higher values indicate more agreement (*windowDiff*$=0$ corresponds to two identical segmentations).

The *S* metric allows comparing trees and taking into account situations when the segmenter places a boundary at a correct position but at a wrong level. *S* is an edit-distance metric: it computes the number of operations necessary to turn one segmentation into another. There are three types of editing operations: add/delete, transpose and substitute (change the level in the tree). The sum is normalized by the number of possible boundaries in the sequence. *S* has an unfortunate downside of being too optimistic, but it allows the breakdown of error types and it explicitly compares trees.

Unlike *windowDiff*, *S* is a similarity metric: higher values correspond to more similar segmentations. The value of *S* between two identical segmentations is 1.

$$S(bs_a, bs_b, n) = \frac{1 - |boundary\_distance(bs_a, bs_b, n)|}{pb(D)} \tag{2}$$

Here $boundary\_distance(bs_a, bs_b, n)$ is the total number of edit operations needed to turn a segmentation bs_a into bs_b, n is the threshold defining the maximum distance of transpositions. $pb(D)$ is the maximum possible number of edits. Segmentations bs_a and bs_a are represented as strings of sets of boundary positions. For example $bs_a = (\{2\}, \{1, 2\}, \{1, 2\})$ corresponds to a hierarchical segmentation of a three-unit sequence in the following manner: a segment boundary at the level 1 after the first unit, segment boundaries at levels 1 and 2 after the second unit and the third unit.

Corpus Analysis. On average, the annotators took 3.5 hours to complete the task ($\sigma = 1.6$). The average depth of the tree is 3.00 levels ($\sigma = 0.65$), suggesting that the annotators prefer shallow but broad structures. Table 1 reports the average breadth of the tree at different levels. In the Table and further in this paper we refer to the bottom level of the tree (i.e., the leaves of the tree or the most fine-grained level of segmentation) as level 1. In Table 1 , level 4 refers to the top level of the tree (the coarsest segmentations). The values were computed using only the breaks explicitly specified by the annotators (i.e. we did not assume that a break at a coarse level implies a break at a more detailed level).

The average breadth of the trees at the bottom (level 1) is lower than that at level 2, indicating that only a small percentage of the entire tree was annotated more then three levels deep. The table also shows the average values of *windowDiff* computed for each possible pair of annotators. The values worsen toward the bottom of the tree, suggesting that the annotators agree more about top-level segments and less and less about finer fluctuations of topic.

We hypothesize that these shallow broad structures are due to the fact that it is difficult for people to create deep recursive structures in their mental representations. However, we do not have any hard data to support this hypothesis. Many of the annotators specifically commented on the difficulty of the task. 9 out of 23 people included comments ranging from notes about specific places to general comments about their lack of confidence. 4 annotators found several (specific) passages they had trouble with.

The average value of pairwise *S* is 0.79. We have noted earlier that the *S* metric tends to be optimistic (that is due to its normalization factor) but it provides a breakdown of disagreements between the annotators. According to *S*, 46.14% of disagreements are errors of omission (some of the annotators did not include segment breaks where other did), 47.56% are disagreements about the level of segmentation (the annotators placed boundaries in the same place but at different levels) and only 6.31% are errors of transposition (the annotators do not agree about the exact placement but place boundaries within 1 position of each other). This distribution is more interesting than the overall value of *S*. Among other things, it shows why it is so important to take into account adjacent levels when evaluating topical trees.

4 The *HAPS* Algorithm

4.1 Factor graphs[1]

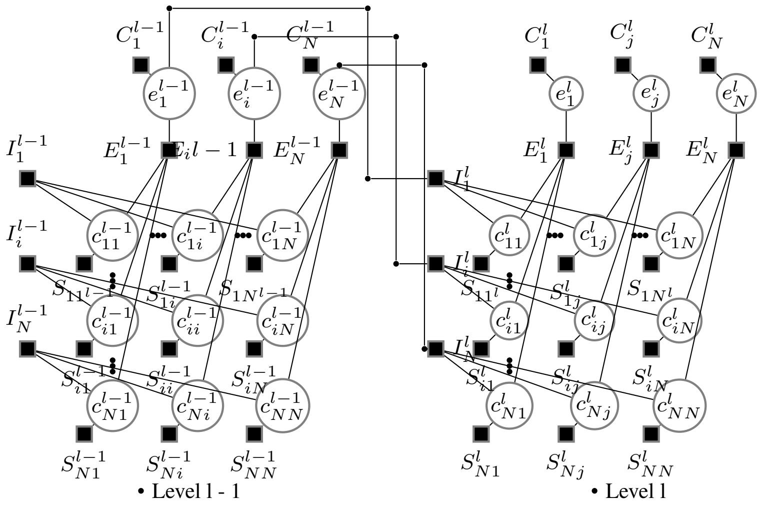
The *HAPS* segmenter is based on the formalism of factor graphs. Factor graphs are a unifying formalism for such graphical models as Markov or Bayesian networks. A factor graph is a bi-partite graph with two types of nodes: *f*actor or function nodes and *v*ariable nodes. Each factor node is connected to those variable nodes which are its arguments. Running the well-known Max-Sum algorithm (Bishop, 2006) on a factor graph finds a configuration of variables which maximizes the sum of all component functions. This is a message-passing algorithm. All variable nodes send messages to their factor neighbours (functions in which those nodes are variables) and all factor nodes send messages to their variable neighbours (their arguments). A message $\mu_{x \to f}$ sent from a variable node x to a function node f is computed as a sum of all incoming messages to x, except the message from the recipient function f:

$$\mu_{x \to f} = \sum_{f' \in N(x) \setminus f} \mu_{f' \to x} \tag{3}$$

[1]The derivation of the *HAPS* algorithm is quite involved. It is unlikely to interest most readers of this paper, so we only present the bare minimum of details about the algorithm, the framework of factor graphs and the derivation of *HAPS* from the underlying model of Affinity Propagation. A detailed account will be made available as supplementary material if the paper is accepted. It will also apear in (Anonymous, 2014).

Table 1: Average breadth of manually created topical trees and *windowDiff* value across different levels

Level	Mean breadth	*windowDiff*
4 (top)	6.53	0.35
3	17.55	0.46
2	17.63	0.47
1 (bottom)	8.80	0.50


(a) Fragment of the factor graph for levels $l-1$ and l

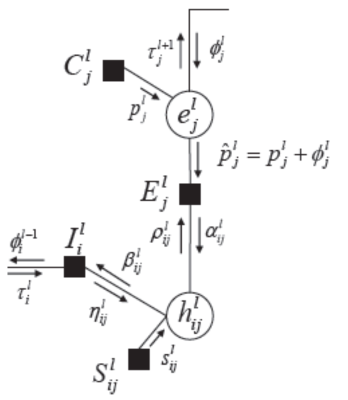

(b) Types of messages sent in the *HAPS* model

Figure 1: Factor graph for *HAPS* – Hierarchical Affinity Propagation for Segmentation

$N(x)$ is the set of all function nodes which are x's neighbours. Intuitively, the message reflects evidence about the distribution of x from all functions which have x as an argument, except the function corresponding to the receiving node f. A message $\mu_{f(x,...)\to x}$ sent from the factor node $f(x,...)$ to the variable node x is computed as a maximum of the value of $f(x)$ plus all messages incoming into $f(x,...)$ other than the message from the recipient node x:

$$\mu_{f\to x} = \max_{N(f)\backslash x}\left(f(x_1,\ldots,x_m) + \sum_{x'\in N(f)\backslash x}\mu_{x'\to f}\right) \tag{4}$$

$N(f)$ is the set of all variable nodes which are f's neighbours. The message reflects the evidence about the distribution of x from function f and its neighbours other than x.

4.2 Hierarchical Affinity Propagation for Segmentation

The goal of this work is to build trees of topical segments. Each segment is characterized by a central point (*SC*) which best describes its content. The objective function is net similarity: the sum of similarities between all *SC*s and the datapoints which they exemplify. The complete sequence of datapoints is to be segmented at each level of the tree, subject to the following constraint: *SC*s at each level l, $l > 1$ must be a subset of the *SC*s from the previous level $l-1$ (in our model, the bottom level is $l = 1$).

Figure 1a shows a fragment of the factor graph describing *HAPS* corresponding to levels l and $l-1$. Overall, the model contains L levels, where $l = L$ is the root level and $l = 1$ is the leaf level. (The superscripts of factor and variable nodes denote the tree level.)

At each level there are N^2 variable nodes c^l_{ij}, where N is the number of datapoints in the sequence we wish to segment. These are binary variables. $c^l_{ij} = 1$ iff, at level l, the datapoint i belongs to the segment centred around datapoint j. Otherwise $c^l_{ij} = 0$. Each level also has N binary variable nodes e^l_j. $e^l_j = 1$ iff there is a segment centred around j at level l.

The are four types of factor nodes in the model in Figure 1a: I, E, C and S. The I factors ensure that each datapoint is assigned to exactly one segment *and* that segment centres at level l are a subset of those from level $l-1$. The E nodes ensure that segments are centred around the segment centres in solid blocks (rather than arbitrary clusters). The I and E factors evaluate to 0 for valid configurations and to $-\infty$ otherwise. The S factors capture similarities between datapoints. $S^l_{ij} = sim(i,j)$ if $c^l_{ij} = 1$. If $c^l_{ij} = 0$ then $S^l_{ij} = 0$.[2] The C factors handle preferences in an analogous manner. Running the Max-Sum algorithm on the factor graph in Figure 1a maximizes the net similarity between all segment centres and their children at all levels:

[2]The value $sim(i,j)$ is specified in the input matrix.

$$\max_{\{c_{ij}^l\},\{e_j^l\}} S(\{c_{ij}^l\},\{e_j^l\}) = \sum_{i,j,l} S_{i,j}^l(c_{ij}^l) + \sum_{i,l} I_i^l(c_{i1}^l,\ldots,c_{iN}^l,e_i^{l-1}) + \sum_{j,l} E_j^l(c_{1j}^l,\ldots,c_{Nj}^l,e_j^l) + \sum_{j,l} C_j^l(e_j^l) \quad (5)$$

Figure 1b shows a close-up view of the messages that need to be sent to find the optimizing configuration of variables. Messages β, η, $\hat{\rho}$ do not need to be sent explicitly: their values are subsumed by other types of messages. We only need to compute explicitly and send four types of messages: α, ρ, ϕ and τ.

Algorithm 1 shows the pseudo-code for the *HAPS* algorithm. Unfortunately it is not possible to include a detailed derivation of the new update messages due to space constraints. Interested readers can find these details in the extended version of this paper available at `http://www.site.uottawa.ca/~ankazant`. The derivation follows the same logic as (Givoni et al., 2011; Kazantseva and Szpakowicz, 2011).

Intuitively, different parts of the update messages in Algorithm 1 correspond to likelihood ratios between two hypotheses: whether a datapoint i *is* or *is not* a part of a segment centred around another datapoint j within a given level l. For example, the availability (α) message sent from a potential segment centre j to the itself within level l is as follows:

$$\alpha_{ij}^l = p_j^l + \phi_j^l + \max_{s=1}^{j}(\sum_{k=s}^{j-1} \rho_{kj}^l) \ + \ \max_{e=j}^{N}(\sum_{k=j+1}^{e} \rho_{kj}^l) \quad (6)$$

Here p_j^l incorporates the information about the preference value for the datapoint j at the level l. ϕ_j^l brings in the information from the coarser level of the tree. The summand $\max_{s=1}^{j}(\sum_{k=s}^{j-1} \rho_{kj}^l)$ encodes the likelihood that there is a segment starting before j given the values of responsibility messages for all datapoints i such that $i < j$ (and hence the information from a more detailed level of the tree as well as the similarities between all datapoints $i,\ i < j$ and j). The summand $\max_{e=j}^{N}(\sum_{k=j+1}^{e} \rho_{kj}^l)$ does the same for the tail end of the segment (all datapoints i such that $i > j$).

Complexity analysis. The *HAPS* model contains N^2 c_{ij}^l nodes at each level. In practice, however, the matrix of similarities $\mathcal{SIM}$ does not need to be fully specified. It is customary to compute this matrix with a large sliding window; the size should be at least twice the anticipated average length. On each iteration, we need to send *L*M*N* messages α and ρ, resulting in *O(L*M*N)* complexity. Here L is the number of levels, N is the number of datapoints in the sequence and M ($M \leq N$) is the size of the sliding window used for computing similarities. Because the computation of ρ and α messages is independent for each row and column respectively, the algorithm would be easy to parallelize.

Parameter settings. An important advantage of *HAPS* is that it does not require the number of segments in advance. Instead, the user needs to set the preference values for each level. However, *HAPS* is fairly resistant to changes in preferences and this generic parameter is a convenient knob for fine-tuning the desired granularity of segmentation, as opposed to specifying the exact number of segments at each level of the tree. In this work we set preferences uniformly, but it is possible to incorporate additional knowledge through more discriminative settings.

In all our experiments the values of preferences are set uniformly for each level of the tree (effectively making all datapoints to be equally likely to be chosen as segment centres within each level). As a starting point, the preference value for the most detailed level of the tree should be about approximately equal to the median similarity value (as specified in the input matrix). A near-zero preference value tends to result in a medium number of segments and is thus suitable to the middle levels of the tree. A negative preference value results in a small number of segments and is appropriate for identifying the most pronounced segment breaks.

5 Experimental Evaluation

In order to evaluate the quality of topical trees produced by *HAPS*, we ran the system on two datasets. We compared the results obtained by *HAPS* against topical trees obtained by iteratively running two high-performance single-level segmenters.

Algorithm 1 Hierarchical Affinity Propagation for Segmentation

```
1: input: 1) $L$ pairwise similarity matrices $\{\mathcal{SIM}^l(i,j)\}_{(i,j)\in\{1,\dots,N\}^2}$; 2) $L$ preferences $p^l$ (one per level $l$) indicating a priori likelihood of point $i$ being a segment centre at level $l$
2: initialization: $\forall i,j : \alpha_{ij} = 0$ (set all availabilities to 0)
3: repeat
4:     iteratively update $\rho$, $\alpha$, $\phi$ and $\tau$ messages
5:     $\forall i,l : \phi_i^{l-1} = \max[0, \alpha_{ii} - \max_{k \neq i}(s_{ik}^l + \alpha_{ik}^l)]$
6:     $\forall i,j,l : \rho_{ij}^l = \begin{cases} \min(0,\tau_i^l) - \max_{k \neq i}(s_{ik}^l + \alpha_{ik}^l) & \text{if } i = j \\ s_{ij}^l + \min[\max(0,-\tau_i^l) - \alpha_{ii}^l, -\max_{k \notin i,j}(s_{ik}^l + \alpha_{ik}^l)] & \text{if } i \neq j \end{cases}$
7:     $\forall i,j,l : \alpha_{ij}^l = \begin{cases} p_j^l + \phi_j^l + \max_{s=1}^{j}(\sum_{k=s}^{j-1}\rho_{kj}^l) + \max_{e=j}^{N}(\sum_{k=j+1}^{e}\rho_{kj}^l) & \text{if } i = j \\ \alpha_{ij,i<j}^l = \min[(\max_{s=1}^{i}\sum_{k=s}^{i-1}\rho_{kj}^l + \sum_{k=i+1}^{j}\rho_{kj}^l + \max_{e=j}^{N}\sum_{k=j+1}^{e}\rho_{kj}^l) + p_j^l + \phi_j^l, \\ \quad \max_{s=1}^{i}\sum_{k=s}^{i-1}\rho_{kj}^l + \min_{s=i+1}^{j}\sum_{k=i+1}^{s-1}\rho_{kj}^l] & \text{if } i < j \\ \min[(\max_{s=1}^{j}\sum_{k=s}^{j-1}\rho_{kj}^l + \sum_{k=j}^{i-1}\rho_{kj}^l + \max_{e=i}^{N}\sum_{k=i+1}^{e}\rho_{kj}^l) + p_j^l + \phi_j^l, \\ \quad \min_{e=j}^{i-1}\sum_{k=e+1}^{i-1}\rho_{kj}^l + \max_{e=i}^{N}\sum_{k=i+1}^{e}\rho_{kj}^l] \end{cases}$
8:     $\forall j,l : \tau_j^{l+1} = p^l(j) + \rho_{jj}^l + \max_{s=1}^{j}(\sum_{k=s}^{j-1}\rho_{kj}^l) + \max_{e=j}^{N}(\sum_{k=j+1}^{e}\rho_{kj}^l)$
9: until convergence
10: compute optimal configuration: $\forall i,j$ $i$ is in the segment centred around $j$ iff $\rho_{ij} + \alpha_{ij} > 0$
11: output: segment centres and segment boundaries
```

Datasets. In these experiments, we used the *Moonstone* dataset described in Section 2 and the Wikipedia dataset compiled by Carroll (2010). The latter dataset was created automatically, using metadata from Web pages. The dataset consists of 66 Wikipedia entries on various topics. In the Wikipedia dataset the annotations (and the results) concern sentences. In the Moonstone dataset we work with paragraphs. To simplify evaluation and interpretation, we produced three-tier trees. This is in line with the average depths of manual annotations in the *Moonstone* dataset.

Baselines. Regrettably, we are not aware of another publicly available hierarchical segmenter. That is why we used as baselines two recent flat segmenters: *MCSeg* (Malioutov and Barzilay, 2006) and *BSeg* (Eisenstein and Barzilay, 2008). Both were first run to produce top-level segmentations. Each segment thus computed was a new input document for segmentation. We repeated the procedure twice to obtain three-tiered trees.

MCSeg cannot be run without knowing the number of segments in advance. Therefore, on each iteration, we had to specify the correct number of segments in the reference segmentation. *BSeg* can be run without specifying the exact number of segments. So we ran it under two settings: with and without knowing the number of segments.

Evaluation metrics. We did our best to obtain a realistic picture of the results, but each metric has its shortcomings. We compared topical trees using *windowDiff* and *evalHDS* (Carroll, 2010). Both metrics are penalties: the higher the values, the worse the hypothetical segmentation. *evalHDS* computes *windowDiff* for each level of the tree in isolation and weighs the errors according to their prominence in the tree. We computed *evalHDS* using the publicly available Python implementation (Carroll, 2010).[3]

When computing *windowDiff*, we treated each level of the tree as a separate segmentation and compared each hypothetical level against a corresponding level in the reference segmentation.

To ensure that evaluations are well-defined at all levels, we propagated the more pronounced reference breaks to lower levels (in both annotations and in the results). This was done to ensure that the whole sequence is segmented at each level. Otherwise *windowDiff* is not well-defined. Conceptually this means that if there is a topical shift of noticeable magnitude (*e.g.,* at the top level), there must be at least a shift of less pronounced magnitude (*e.g.,* an an intermediate level).

The *Moonstone* dataset has on average 4.8 annotations per chapter. It is not obvious how to combine these multiple annotations. We evaluated each hypothetical segmentation against each available gold standard separately. We report the averages across all annotators – for both *evalHDS* and *windowDiff* – per level.

Preprocessing. The representations used by *HAPS* and the *MCSeg* are very similar. Both systems compute a matrix of similarities between atomic units of the document (sentences or paragraphs). Each unit was represented as a bag of words. The vectors were further weighted by the *tf.idf* value of the term and also smoothed (in the same manner as in (Malioutov and Barzilay, 2006)). We computed similarity between vectors corresponding to each sentence or paragraph using cosine similarity. We used tenfold cross-validation on the Wikipedia dataset and fourfold cross-validation on the smaller *Moonstone* dataset.

Evaluating the quality of the *SC*s. In addition to finding topical shifts, *HAPS* identifies *SC*s – sentences or paragraphs which best capture what each segment is about. In order to get a rough estimate of the quality of the *SC*s, we extracted paragraphs identified as segment centres at the second (middle) level of *HAPS* trees. These pseudo-summaries were then compared to summaries created by an automatic summarizer *CohSum*. We used ROUGE-1 and ROUGE-L metrics (Lin, 2004) as a basis for comparison. *CohSum* identifies the most salient sentences in a document by running a variant of the TextRank algorithm (Mihalcea and Tarau, 2004) on the entire document. In addition to using lexical similarity, the summarizer takes into account coreference links between sentences. We ran *CohSum* at 10% compression rate.

The summarization experiment was performed on the *Moonstone* corpus. We also collected 20 chapters from several other XIX century novels and used it in a separate experiment. The ROUGE package requires manually written summaries to compare with the automatically created ones. We obtained the summaries from the SparkNotes website.[4]

6 Results and Discussion

Table 2 shows the results of comparing *HAPS* with two baseline segmenters using *windowDiff* and *evalHDS*. *HAPS* was run without knowing the number of segments. *MCSeg* required that the exact number be specified. *BSeg* was tested with and without that parameter. Therefore, rows 3 and 4 in Table 2 correspond to baselines considerably more informed than *HAPS*. This is especially true about the bottom levels where sometimes knowing the exact number of segments unambiguously determines the only possible segmentation.

[3]When working with the *Moonstone* dataset, we realized that the software produces very low values, almost too good to be true. That is because the bottommost annotations are very fine-grained. Sometimes each paragraph corresponds to a separate segment. This causes problems for the software. So, when we report *evalHDS* values for the *Moonstone* dataset, we only consider two top levels of the tree, disregarding the leaves. We also remove the "too good to be true" outliers, though the "bad" tail is left intact. We applied the same procedure to all three segmenters, only for the *Moonstone* dataset.

[4]www.sparknotes.com/

	Level	*Moonstone windowDiff*	*Wikipedia windowDiff*	*Moonstone evalHDS*	*Wikipedia evalHDS*
HAPS	3 (top)	0.337 (± 0.060)	0.421 (± 0.060)	0.353	0.450
	2 (middle)	0.422 (± 0.060)	0.447 (± 0.070)	(± 0.072)	(± 0.015)
	1 (bottom)	0.556 (± 0.070)	0.617 (± 0.080)		
MinCutSeg-iter.	3 (top)	0.375	0.440 (± 0.075)	0.377	0.444
	2 (middle)	0.541	0.424 (± 0.064)	(± 0.002)	(± 0.002)
segm. known	1 (bottom)	0.601	0.471 (± 0.057)		
BayesSeg-iter.	3 (top)	0.353 (± 0.071)	0.391 (± 0.070)	0.367	0.370
	2 (middle)	0.406 (± 0.053)	0.344 (± 0.033)	(± 0.089)	(± 0.019)
segm. known	1 (bottom)	0.504 (± 0.064)	0.354 (± 0.033)		
BayesSeg-iter.	3 (top)	0.600 (± 0.071)	0.637 (± 0.070)	0.453	0.437
	2 (middle)	0.447 (± 0.053)	0.877 (± 0.033)	(± 0.089)	(± 0.022)
segm. unknown	1 (bottom)	0.545 (± 0.064)	0.952 (± 0.033)		

Table 2: Evaluation of *HAPS* and iterative versions of *APS*, *MCSeg* and *BSeg* using *windowDiff* per level (mean *windowDiff* and standard deviation for cross-validation)

	Moonstone corpus		Austen corpus	
	ROUGE-1	ROUGE-L	ROUGE-1	ROUGE-L
Segment centres	0.341 (0.312, 0.370)	0.321 (0.298, 0.346)	0.291 (0.272, 0.311)	0.301 (0.293, 0.330)
CohSum summaries	0.294 (0.243, 0.334)	0.269 (0.226, 0.306)	0.305 (0.290, 0.320)	0.307 (0.287, 0.327)

Table 3: *HAPS* segment centres compared to *CohSum* summaries: ROUGE scores and 95% confidence intervals

The results suggest that *HAPS* performs well on the *Moonstone* dataset even when compared to more informed baselines. This applies to both metrics, *windowDiff* and *evalHDS*. *BSeg* performs slightly better at the bottom levels of the tree when it has the information about the exact number of segments. We hypothesize that the advantage may be due to this additional information, especially when segmenting already small segments at level 1 into a predefined number of segments. Another explanation may be that when using *windowDiff* as the evaluation metric, *HAPS* was fine-tuned so as to maximize the value of *windowDiff* at the top level, effectively disregarding lower levels of segmentation.

On the Wikipedia dataset all segmenters perform worse. Using that scale, informed *BSeg* performs the best, but it is interesting to note a significant drop in performance when the number of segments is not specified.

Overall *HAPS* appears to perform better than, or comparably to, the more informed baselines, and much better than the baseline not given information about the number of segments.

We also made a preliminary attempt to evaluate the quality of *SC*s by comparing them to the summaries created by the *CohSum* summarizer. In addition to working with the *Moonstone* corpus, we collected a corpus of 20 chapters from various novels by Jane Austen.

Table 3 shows the results. They are not conclusive because there is no evidence that ROUGE scores correlate with the quality of automatically created summaries for literature. However, according to the scores in Table 3 the summaries created by *CohSum* cannot be distinguished from simple summaries composed of *SC*s identified by *HAPS*. We interpret this as a sign that the *SC*s identified by *HAPS* are approximately as informative as those created by an automatic summarizer.

7 Conclusions

In this paper we presented *HAPS*, a hierarchical segmenter for free text. Given an input document, *HAPS* creates a topical tree and for each segment it identifies a segment centre. One of the advantages of *HAPS* is that it does not require knowing the exact number of segments in advance but instead it estimates it given information about generic preferences about granularity of segmentation. We also created a corpus of hierarchical segmentations that has been annotated by 3-6 people per chapter.

A Java implementation of *HAPS* and the *Moonstone* corpus will be made publicly available.

Acknowledgements

The authors would like to thank Chris Fournier (for computing *S* values using a beta version of SegEval software for hierarchical datasets), Lucien Carrol (for help and discussion of the *evalHDS* software and representation) and Christian Smith (for allowing us to use his implementation of CohSum).

References

Anonymous. 2014. *Anonymized Title*. Ph.D. thesis, Anonymous University.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

David Blei and Pedro Moreno. 2001. Topic segmentation with an aspect hidden Markov Model. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 343–348.

Lucien Carroll. 2010. Evaluating Hierarchical Discourse Segmentation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 993–1001.

Lan Du, Wray Buntine, and Mark Johnson. 2013. Topic Segmentation with a Structured Topic Model. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200, Atlanta, Georgia.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian Unsupervised Topic Segmentation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 334–343, Honolulu, Hawaii.

Jacob Eisenstein. 2009. Hierarchical Text Segmentation from Multi-Scale Lexical Cohesion. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 353–361. The Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level Discourse Parsing with Rich Linguistic Features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 60–68, Jeju Island, Korea, July. Association for Computational Linguistics.

Chris Fournier and Diana Inkpen. 2012. Segmentation Similarity and Agreement. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 152–161, Montréal, Canada.

Inmar E. Givoni, Clement Chung, and Brendan J. Frey. 2011. Hierarchical Affinity Propagation. In *Uncertainty in AI, Proceedings of the Twenty-Seventh Conference (2011)*, pages 238–246.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring Content Models for Multi-Document Summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado, June.

Marti A. Hearst. 1997. TextTiling: segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Hugo Hernault, Helmut Prendinger, David A. duVerlea, and Mitsuru Ishizuka. 2010. HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 3:1–33.

Anna Kazantseva and Stan Szpakowicz. 2011. Linear Text Segmentation Using Affinity Propagation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 284–293, Edinburgh, Scotland.

Anna Kazantseva and Stan Szpakowicz. 2012. Topical Segmentation: a Study of Human Performance and a New Measure of Quality. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 211–220, Montréal, Canada.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of summaries. In *Text Summarization Branches Out, Proceedings of the ACL Workshop*, pages 74–81, Barcelona, Spain.

Igor Malioutov and Regina Barzilay. 2006. Minimum Cut Model for Spoken Lecture Segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 25–32, Sydney, Australia.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Meghana Marathe. 2010. Lexical Chains Using Distributional Measures of Concept Distance. Master's thesis, University of Toronto.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, Mass.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In Dekang Lin and Dekai Wu, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing 2004*, pages 404–411, Barcelona, Spain.

Hemant Misra, François Yvon, Olivier Cappé, and Joemon M. Jose. 2011. Text segmentation: A topic modeling perspective. *Information Processing and Management*, 47(4):528–544.

Hyo-Jung Oh, Sung Hyon Myaeng, and Myung-Gil Jang. 2007. Semantic passage segmentation based on sentence topics for question answering. *Information Sciences, an International Journal*, 177:3696–3717.

Lev Pevzner and Marti A. Hearst. 2002. A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*, 28(1):19–36.

Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia.

Christian Smith, Henrik Danielsson, and Arne Jnsson. 2012. A more cohesive summarizer. In *24th International Conference on Computational Linguistics, Proceedings of COLING 2012: Posters*, pages 1161–1170, Mumbai, India.

Fei Song, William M. Darling, Adnan Duric, and Fred W. Kroon. 2011. An iterative approach to text segmentation. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, ECIR'11, pages 629–640, Berlin, Heidelberg. Springer-Verlag.

Florian Wolf and Edward Gibson. 2006. *Coherence in Natural Language: Data Structures and Applications*. MIT Press, Cambridge, MA.

Yaakov Yaari. 1997. Segmentation of Expository Texts by Hierarchical Agglomerative Clustering. In *Proceedings of International Conference on Recent Advances in Natural Language Processing RANLP97*, pages 59–65, Tzigov Chark, Bulgaria.

Gilbert Youmans. 1991. A new tool for discourse analysis: The vocabulary-management profile. *Language*, 67(4):763–789.