Unsupervised Text Segmentation Using Semantic Relatedness Graphs

Goran Glavaš, Federico Nanni, Simone Paolo Ponzetto
Data and Web Science Group
University of Mannheim
B6 26, DE-68161 Mannheim, Germany
{goran,federico,simone}@.informatik.uni-mannheim.de

Abstract

Segmenting text into semantically coherent fragments improves readability of text and facilitates tasks like text summarization and passage retrieval. In this paper, we present a novel unsupervised algorithm for linear text segmentation (TS) that exploits word embeddings and a measure of semantic relatedness of short texts to construct a *semantic relatedness graph* of the document. Semantically coherent segments are then derived from maximal cliques of the relatedness graph. The algorithm performs competitively on a standard synthetic dataset and outperforms the best-performing method on a real-world (i.e., non-artificial) dataset of political manifestos.

1 Introduction

Despite the fact that in mainstream natural language processing (NLP) and information retrieval (IR) texts are modeled as bags of unordered words, texts are sequences of semantically coherent segments, designed (often very thoughtfully) to ease readability and understanding of the ideas conveyed by the authors. Although authors may explicitly define coherent segments (e.g., as paragraphs), many texts, especially on the web, lack any explicit segmentation.

Linear text segmentation aims to represent texts as sequences of semantically coherent segments. Besides improving readability and understandability of texts for readers, automated text segmentation is beneficial for NLP and IR tasks such as text summarization (Angheluta et al., 2002; Dias et al., 2007) and passage retrieval (Huang et al., 2003; Dias et al., 2007). Whereas early approaches to unsupervised text segmentation measured the coherence of segments via raw term overlaps between sentences (Hearst, 1997; Choi, 2000), more recent methods (Misra et al., 2009; Riedl and Biemann, 2012) addressed the issue of sparsity of term-based representations by replacing term-vectors with vectors of latent topics.

A topical representation of text is, however, merely a vague approximation of its meaning. Considering that the goal of TS is to identify *semantically* coherent segments, we propose a TS algorithm aiming to directly capture the semantic relatedness between segments, instead of approximating it via topical similarity. We employ word embeddings (Mikolov et al., 2013) and a measure of semantic relatedness of short texts (Šarić et al., 2012) to construct a relatedness graph of the text in which nodes denote sentences and edges are added between semantically related sentences. We then derive segments using the maximal cliques of such similarity graphs.

The proposed algorithm displays competitive performance on the artifically-generated benchmark TS dataset (Choi, 2000) and, more importantly, outperforms the best-performing topic modeling-based TS method on a real-world dataset of political manifestos.

2 Related Work

Automated text segmentation received a lot of attention in NLP and IR communities due to its usefulness for text summarization and text indexing. Text segmentation can be performed in two different ways, namely (1) with the goal of obtaining linear segmentations (i.e. detecting the sequence of different segments in a text) , or (2) in order to obtain hierarchical segmentations (i.e. defining a structure of subtopics between the detected segments). Like the majority of TS methods (Hearst, 1994; Brants et al., 2002; Misra et al., 2009; Riedl

*Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics (*SEM 2016)*, pages 125–130,
Berlin, Germany, August 11-12, 2016.

and Biemann, 2012), in this work we focus on linear segmentation of text, but there is also a solid body of work on hierarchical TS, where each top-level segment is further broken down (Yaari, 1997; Eisenstein, 2009).

Hearst (1994) introduced TextTiling, one of the first unsupervised algorithms for linear text segmentation. She exploits the fact that words tend to be repeated in coherent segments and measures the similarity between paragraphs by comparing their sparse term-vectors. Choi (2000) introduced the probabilistic algorithm using matrix-based ranking and clustering to determine similarities between segments. Galley et al. (2003) combined content-based information with acoustic cues in order to detect discourse shifts whereas Utiyama and Isahara (2001) and Fragkou et al. (2004) minimized different segmentation cost functions with dynamic programming.

The first segmentation approach based on topic modeling (Brants et al., 2002) employed the probabilistic latent semantic analysis (pLSA) to derive latent representations of segments and determined the segmentation based on similarities of segments' latent vectors. More recent models (Misra et al., 2009; Riedl and Biemann, 2012) employed the latent Dirichlet allocation (LDA) (Blei et al., 2003) to compute the latent topics and displayed superior performance to previous models on standard synthetic datasets (Choi, 2000; Galley et al., 2003). Misra et al. (2009) used dynamic programming to find globally optimal segmentation over the set of LDA-based segment representations, whereas Riedl and Biemann (2012) introduced TopicTiling, an LDA-driven extension of Hearst's TextTiling algorithm where segments are, represented as dense vectors of dominant topics of terms they contain (instead of as sparse term vectors). Riedl and Biemann (2012) show that TopicTiling outperforms at-that-time state-of-the-art methods for unsupervised linear segmentation (Choi, 2000; Utiyama and Isahara, 2001; Galley et al., 2003; Fragkou et al., 2004; Misra et al., 2009) and that it is also faster than other LDA-based methods (Misra et al., 2009).

In the most closely related work to ours, Malioutov and Barzilay (2006) proposed a graph-based TS approach in which they first construct the fully connected graph of sentences, with edges weighted via the cosine similarity between bag-of-words sentence vectors, and then run the minimum normalized multiway cut algorithm to obtain the segments. Similarly, Ferret (2007) builds the similarity graph, only between words instead of between sentences, using sparse co-occurrence vectors as semantic representations for words. He then identifies topics by clustering the word similarity graph via the Shared Nearest Neighbor algorithm (Ertöz et al., 2004). Unlike these works, we use the dense semantic representations of words and sentences (i.e., embeddings), which have been shown to outperform sparse semantic vectors on a range of NLP tasks. Also, instead of looking for minimal cuts in the relatedness graph, we exploit the maximal cliques of the relatedness graph between sentences to obtain the topic segments.

3 Text Segmentation Algorithm

Our TS algorithm, dubbed GRAPHSEG, builds a *semantic relatedness graph* in which nodes denote sentences and edges are created for pairs of semantically related sentences. We then determine the coherent segments by finding maximal cliques of the relatedness graph. The novelty of GRAPHSEG is in the fact that it directly exploits the semantics of text instead of approximating the meaning with topicality.

3.1 Semantic Relatedness of Sentences

The measure of semantic relatedness between sentences we use is an extension of a salient *greedy lemma alignment* feature proposed in a supervised model by Šarić et al. (2012). They greedily align content words between sentences by the similarity of their distributional vectors and then sum the similarity scores of aligned word pairs. However, such greedily obtained alignment is not necessarily optimal. In contrast, we compute the optimal alignment by (1) creating a weighted complete bipartite graph between the sets of content words of the two sentences (i.e., each word from one sentence is connected with a relatedness edge to all of the words in the other sentence) and (2) running a bipartite graph matching algorithm known as the Hungarian method (Kuhn, 1955) that has the polynomial complexity. The similarities of content words between sentences (i.e., the weights of the bipartite graph) are computed as the cosine of the angle between their corresponding embedding vectors (Mikolov et al., 2013).

Let A be the set of word pairs in the optimal alignment between the content-word sets of the two

sentences S_1 and S_2, i.e., $A = \{(w_1, w_2) \mid w_1 \in S_1 \wedge w_2 \in S_2\}$. We then compute the semantic relatedness for two given sentences S_1 and S_2 as follows:

$$sr(S_1, S_2) = \sum_{(w_1,w_2)\in A} \cos(v_1, v_2) \cdot \min(ic(w_1), ic(w_2))$$

where v_i is the embedding vector of the word w_i and $ic(w)$ is the information content (IC) of the word w, computed based on the relative frequency of w in some large corpus C:

$$ic(w) = -\log \frac{freq(w) + 1}{|C| + \sum_{w' \in C} freq(w')}.$$

We utilize the IC weighting of embedding similarity because we assume that matches between less frequent words (e.g., *guitar* and *ukulele*) contribute more to sentence relatedness than pairs of similar but frequent words (e.g., *do* and *make*). We used Google Books Ngrams (Michel et al., 2011) as a large corpus C for estimating relative frequencies of words in a language.

Because there will be more aligned pairs between longer sentences, the relatedness score will be larger for longer sentences merely because of their length (regardless of their actual similarity). Thus, we normalize the $sr(S_1, S_2)$ score first with the length of S_1 and then with the length S_2 and we finally average these two normalized scores:

$$rel(S_1, S_2) = \frac{1}{2} \cdot \left(\frac{sr(S_1, S_2)}{|S_1|} + \frac{sr(S_1, S_2)}{|S_2|} \right).$$

3.2 Graph-Based Segmentation

All sentences in a text become nodes of the relatedness graph G. We then compute the semantic similarity, as described in the previous subsection, between all pairs of sentences in a given document. For each pair of sentences for which the semantic relatedness is above some treshold value τ we add an edge between the corresponding nodes of G. Next, we employ the Bron-Kerbosch algorithm (Bron and Kerbosch, 1973) to compute the set $\mathcal{Q}$ of all maximal cliques of G. We then create the initial set of segments SG by merging adjacent sentences found in at least one maximal clique $Q \in \mathcal{Q}$ of graph G. Next, we merge the adjacent segments sg_i and sg_{i+1} for which there is at least one clique $Q \in \mathcal{Q}$ containing at least one sentence from sg_i and one sentence from sg_{i+1}. Finally, given the minimal segment size n, we merge segments sg_i with less than n sentences with the semantically more related of the two adjacent segments – sg_{i-1} or sg_{i+1}. The relatedness between two adjacent segments ($sgr(sg_i, sg_{i+1})$) is computed as the average relatedness between their respective sentences:

$$sgr(SG_1, SG_2) = \frac{1}{|SG_1||SG_2|} \sum_{\substack{S_1 \in SG_1 \\ S_2 \in SG_2}} rel(S_1, S_2).$$

We exemplify the creation of segments from maximal cliques in Table 1. The complete segmentation algorithm is fleshed out in Algorithm 1.[1]

Step	Sets
Cliques $\mathcal{Q}$	{1, 2, 6}, {2, 4, 7}, {3, 4, 5}, {1, 8, 9}
Init. seg.	{1, 2}, {3, 4, 5}, {6}, {7} {8, 9}
Merge seg.	{1, 2, 3, 4, 5}, {6}, {7}, {8, 9}
Merge small	{1, 2, 3, 4, 5}, {6, 7}, {8, 9}

Table 1: Creating segments from graph cliques ($n = 2$). In the third step we merge segments {1, 2, 3} and {4, 5} because the second clique contains sentences 2 (from the left segment) and 4 (from the right segment). In the final step we merge single sentence segments (assuming $segs(\{1,2,3,4,5\},\{6\}) < segs(\{6\},\{7\})$ and $segs(\{7\},\{8,9\}) < segs(\{6\},\{7\})$).

4 Evaluation

In this section, we first introduce the two evaluation datasets that we use one being the commonly used synthetic dataset and the other a realistic dataset of politi- cal manifestos. Following, we present the experimental setting and finally describe and discuss the results achieved by our GRAPHSEG algorithm and how it compares to other TS models.

4.1 Datasets

Unsupervised methods for text segmentation have most often been evaluated on synthetic datasets with segments from different sources being concatenated in artificial documents (Choi, 2000; Galley et al., 2003). Segmenting such artificial texts is easier than segmenting real-world documents. This is why besides on the artificial Choi dataset we also evaluate GRAPHSEG on a real-world dataset of political texts from the Manifesto Project,[2,3] manually

[1] We make the GraphSeg tool freely available at the following address: `https://gg42554@bitbucket.org/gg42554/graphseg.git`

[2] `https://manifestoproject.wzb.eu`

[3] We used the set of six documents manifestos – three Republican and three Democrat manifestos from the 2004,

Algorithm 1: *Segment*($text$, τ, n)

$G \leftarrow (V \leftarrow \varnothing, E \leftarrow \varnothing)$
$\mathcal{S} \leftarrow sentences(text)$
$SG \leftarrow \varnothing$
// constructing the similarity graph
for each sentence $S_i \in \mathcal{S}$ **do**
  $V \leftarrow V \cup \{S_i\}$
for each pair $(S_i, S_j) \mid S_i, S_j \in \mathcal{S}$ **do**
  if $rel(S_i, S_j) > \tau$ **do**
    $E \leftarrow E \cup (\{S_i\}, \{S_j\})$
// creating initial segments from cliques
$\mathcal{Q} \leftarrow cliques(G)$
for each clique $Q \in \mathcal{Q}$ **do**
  for each $(S_i, S_j), S_i, S_j \in Q$ **do**
    if $j - i = 1$ **do**
      if $sg(S_i) = \varnothing$ **and** $sg(S_j) = \varnothing$ **do**
        $SG \leftarrow SG \cup \{S_i, S_j\}$
      elif $sg(S_i) \neq \varnothing$ **and** $sg(S_j) = \varnothing$ **do**
        $sg(S_i) \leftarrow sg(S_i) \cup \{S_j\}$
      elif $sg(S_i) = \varnothing$ **and** $sg(S_j) \neq \varnothing$ **do**
        $sg(S_j) \leftarrow sg(S_j) \cup \{S_i\}$
// merging adjacent segments
for each segment $sg_i \in SG$ **do**
  if $\exists Q \in \mathcal{Q} \mid (\exists S_j, S_k \in Q \mid S_j \in sg_i \wedge S_k \in sg_{i+1})$ **do**
    $SG \leftarrow SG \setminus \{sg_i, sg_{i+1}\}$
    $SG \leftarrow SG \cup (sg_i \cup sg_{i+1})$
// merging too small segments
for each segment $sg_i \in SG$ **do**
  if $|sg_i| < n$ **do**
    if $sgr(sg_{i-1}, sg_i) > sgr(sg_i, sg_{i+1})$ **do**
      $SG \leftarrow SG \setminus \{sg_{i-1}, sg_i\}$
      $SG \leftarrow SG \cup (sg_{i-1} \cup sg_i)$
    else do
      $SG \leftarrow SG \setminus \{sg_i, sg_{i+1}\}$
      $SG \leftarrow SG \cup (sg_i \cup sg_{i+1})$
return SG

labeled by domain experts with segments of seven different topics (e.g., economy and welfare, quality of life, foreign affairs). The selected manifestos contain between 1000 and 2500 sentences, with segments ranging in length from 1 to 78 sentences, which is in sharp contrast to the Choi dataset where all segments are of similar size.

4.2 Experimental Setting

To allow for comparison with previous work, we evaluate GRAPHSEG on four subsets of the Choi dataset, differing in number of sentences the segments contain. For the evaluation on the Choi dataset, the GRAPHSEG algorithm made use of the publicly available word embeddings built from a Google News dataset.[4]

Both LDA-based models (Misra et al., 2009; Riedl and Biemann, 2012) and GRAPHSEG rely on corpus-derived word representations. Thus, we evaluated on the Manifesto dataset both the domain-adapted and domain-unadapted variants of these methods. The domain-adapted variants of the models used the unlabeled domain corpus – a test set of 466 unlabeled political manifestos – to train the domain-specific word representations. This means that we obtain (1) in-domain topics for the LDA-based TopicTiling model of Riedl and Biemann (2012) and (2) domain-specific embeddings for the GRAPHSEG algorithm. On the Manifesto dataset we also evaluate a baseline that randomly (50% chance) starts a new segment at points m sentences apart, with m being set to half of the average length of gold segments.

We evaluate the performance using two standard TS evaluation metrics – P_k (Beeferman et al., 1999) and WindowDiff (WD) (Pevzner and Hearst, 2002). P_k is the probability that two randomly drawn sentences mutually k sentences apart are classified incorrectly – either as belonging to the same segment when they are in different gold segments or as being in different segments when they are in the same gold segment. Following Riedl and Biemann (2012), we set k to half of the document length divided by the number of gold segments. WindowDiff is a stricter version of P_k as, instead of only checking if the randomly chosen sentences are in the same predicted segment or not, it compares the exact number of segments between the sentences in the predicted segmentation with the number of segments in between the same sentences in the gold standard. Lower scores indicate better performance for both these metrics.

The GRAPHSEG algorithm has two parameters: (1) the sentence similarity treshold τ which is used when creating edges of the sentence relatedness graph and (2) the minimal segment size n, which we utilize to merge adjacent segments that are too small. In all experiments we use grid-search in a folded cross-validation setting to jointly optimize both parameters. In view of comparison with other models, the parameter optimization is justified be-

2008, and 2012 U.S. elections

[4] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing

Method	3-5		6-8		9-11		3-11	
	P_k	WD	P_k	WD	P_k	WD	P_k	WD
Choi (2000)	12.0	–	9.0	–	9.0	–	12.0	–
Brants et al. (2002)	7.4	–	8.0	–	6.8	–	10.7	–
Fragkou et al. (2004)	5.5	–	3.0	–	1.3	–	7.0	–
Misra et al. (2009)	23.0	–	15.8	–	14.4	–	16.1	–
GRAPHSEG	5.6	8.7	7.2	9.4	6.6	9.6	7.2	9.0
Misra et al. (2009)*	2.2	–	2.3	–	4.1	–	2.3	–
Riedl and Biemann (2012)*	1.2	1.3	0.8	0.9	0.6	0.7	1.0	1.1

Table 2: Performance on different portions of the Choi dataset (*with domain-adapted topic model).

Method	P_k	WD
Random baseline	40.60	49.17
Riedl and Biemann (2012)	33.39	38.31
GRAPHSEG	28.09	34.04
Riedl and Biemann (2012)*	32.94	37.59
GRAPHSEG*	28.08	34.00

Table 3: Performance on the Manifesto dataset (*domain-adapted variant).

cause other models, e.g., TopicTiling (Riedl and Biemann, 2012), also have parameters (e.g., number of topics for the topic model) which are optimized using cross-validation.

4.3 Results and Discussion

In Table 2 we report the performance of GRAPHSEG and prominent TS methods on the synthetic Choi dataset. GRAPHSEG performs competitively, outperforming all methods but (Fragkou et al., 2004) and domain-adapted versions of LDA-based models (Misra et al., 2009; Riedl and Biemann, 2012). However, the approach by (Fragkou et al., 2004) uses the gold standard information – the average gold segment size – as input. On the other hand, the LDA-based models adapt their topic models on parts of the Choi dataset itself. Despite the fact that they use different documents for training the topic models from those used for evaluating segmentation quality, the evaluation is still tainted because snippets from the original documents appear in multiple artificial documents – some of which belong to the the training set and others to the test set, as admitted by Riedl and Biemann (2012) and this is why their reported performance on this dataset is overestimated.

In Table 3 we report the results on the Manifesto dataset. Results of both TopicTiling and GRAPHSEG indicate that the realistic Manifesto dataset is much more difficult to segment than the artificial Choi dataset. The GRAPHSEG algorithm significantly outperforms the TopicTiling method ($p < 0.05$, Student's t-test). In-domain training of word representations, topics for TopicTiling and word embeddings for *GraphSeg*, does not significantly improve the performance for neither of the two models. This result contrasts previous findings (Misra et al., 2009; Riedl and Biemann, 2012) in which the performance boost was credited to the in-domain trained topics and supports our hypothesis that the performance boost of the LDA-based methods' with in-domain trained topics originates from information leakage between different portions of the synthetic Choi dataset.

5 Conclusion

In this work we presented GRAPHSEG, a novel graph-based algorithm for unsupervised text segmentation. GRAPHSEG employs word embeddings and extends a measure of semantic relatedness to construct a relatedness graph with edges established between semantically related sentences. The segmentation is then determined by the maximal cliques of the relatedness graph and improved by semantic comparison of adjacent segments.

GRAPHSEG displays competitive performance compared to best-performing LDA-based methods on a synthetic dataset. However, we identify and discuss evaluation issues pertaining to LDA-based TS on this dataset. We also performed an evaluation on the real-world dataset of political manifestos and showed that in a realistic setting GRAPHSEG significantly outperforms the state-of-the-art LDA-based TS model.

Acknowledgments

We thank the Manifesto Project researchers for making the topically annotated manifestos freely available for research purposes. We thank the anonymous reviewers for their useful comments.

References

Roxana Angheluta, Rik De Busser, and Marie-Francine Moens. 2002. The use of topic segmentation for automatic summarization. In *Proceedings of the ACL-2002 Workshop on Automatic Summarization*, pages 11–12.

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine learning*, 34(1-3):177–210.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Thorsten Brants, Francine Chen, and Ioannis Tsochantaridis. 2002. Topic-based document segmentation with probabilistic latent semantic analysis. In *Proceedings of CIKM*, pages 211–218. ACM.

Coen Bron and Joep Kerbosch. 1973. Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577.

Freddy YY Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of NAACL*, pages 26–33. Association for Computational Linguistics.

Gaël Dias, Elsa Alves, and José Gabriel Pereira Lopes. 2007. Topic segmentation algorithms for text summarization and passage retrieval: An exhaustive evaluation. In *AAAI*, volume 7, pages 1334–1339.

Jacob Eisenstein. 2009. Hierarchical text segmentation from multi-scale lexical cohesion. In *Proceedings of HLT-NAACL*, pages 353–361. Association for Computational Linguistics.

Levent Ertöz, Michael Steinbach, and Vipin Kumar. 2004. Finding topics in collections of documents: A shared nearest neighbor approach. *Clustering and Information Retrieval*, pages 83–103.

Olivier Ferret. 2007. Finding document topics for improving topic segmentation. In *ACL*, volume 7, pages 480–487. Citeseer.

Pavlina Fragkou, Vassilios Petridis, and Ath Kehagias. 2004. A dynamic programming algorithm for linear text segmentation. *Journal of Intelligent Information Systems*, 23(2):179–197.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of ACL*, pages 562–569. Association for Computational Linguistics.

Marti A Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of the ACL*, pages 9–16. Association for Computational Linguistics.

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

Xiangji Huang, Fuchun Peng, Dale Schuurmans, Nick Cercone, and Stephen E Robertson. 2003. Applying machine learning to text segmentation for information retrieval. *Information Retrieval*, 6(3-4):333–362.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of COLING-ACL*, pages 25–32. Association for Computational Linguistics.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Aiden Erez Lieberman. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Hemant Misra, François Yvon, Joemon M Jose, and Olivier Cappe. 2009. Text segmentation via topic modeling: An analytical study. In *Proceedings of CIKM*, pages 1553–1556. ACM.

Lev Pevzner and Marti A Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Martin Riedl and Chris Biemann. 2012. TopicTiling: A text segmentation algorithm based on LDA. In *Proceedings of ACL 2012 Student Research Workshop*, pages 37–42. Association for Computational Linguistics.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. TakeLab: Systems for measuring semantic text similarity. In *Proceedings of SemEval*, pages 441–448. Association for Computational Linguistics.

Masao Utiyama and Hitoshi Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of ACL*, pages 499–506. Association for Computational Linguistics.

Yaakov Yaari. 1997. Segmentation of expository texts by hierarchical agglomerative clustering. In *Proceedings of RANLP*.