A Tool for Efficient Content Compilation

Boris Galitsky
Knowledge-Trail Inc
San Jose CA USA
bgalitsky@hotmail.com

Abstract

We build a tool to assist in content creation by mining the web for information relevant to a given topic. This tool imitates the process of essay writing by humans: searching for topics on the web, selecting content fragments from the found document, and then compiling these fragments to obtain a coherent text. The process of writing starts with automated building of a table of content by obtaining the list of key entities for the given topic extracted from web resources such as Wikipedia. Once a table of content is formed, each item forms a seed for web mining. The tool builds a full-featured structured Word document with table of content, section structure, images and captions and web references for all included text fragments.

Two linguistic technologies are employed: for relevance verification, we use similarity computed as a tree similarity between parse trees for a seed and candidate text fragment. For text coherence, we use a measure of agreement between a given and consecutive paragraph by tree kernel learning of their discourse trees.

The tool is available at http://animatronica.io/submit.html.

1 Introducing content compilation problem

In the modern society, writing and creating content is one of the most frequent human activity. An army of content creators, from students to professional writers produce various kinds of documents for various audiences. Not all of these documents are expected to be innovative, break-through or extremely important. The target of the tool being proposed is assistance with routine document creation process (Fig. 1) where most information is available on the web and needs to be collected, integrated and properly referenced.

A number of content generation software systems are available in specific business domains (Johnson 2016). Most of content generation software are template-based which limits their efficiency and volume of produced content (Hendrikx et al 2015). An interesting class of content generation system is based on verbalizing some numerical data. Also, content generation for computer game support turned out to be fruitful (Liapis et al 2013). Deep-learning – based generation of a sequence of words has a limited applicability for large scale content production industrial systems. The goal of this study is to build a content compilation assistance system that would meet the following criteria:

- Produces high volume cohesive text on a given topic in a domain-independent manner;
- Collects text fragments from the web and combines them to assist in research on a given topic, provide systematic references;
- Combines text, image and video resources in the resultant document;
- Suitable for producing a final report and manual editing by students, researchers in various fields in science, engineering, business and law.

On the bottom-left of Fig. 1 we show the main problem that needs to be solved to build a document from fragments collected from the web. For given two fragments, we need to determine if one can reasonably follow another in a cohesive manner. W build a discourse representation for each fragment an learn this representation to classify a pair of consecutive paragraphs as cohesive or not.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pages 198–202, Osaka, Japan, December 11-17 2016.

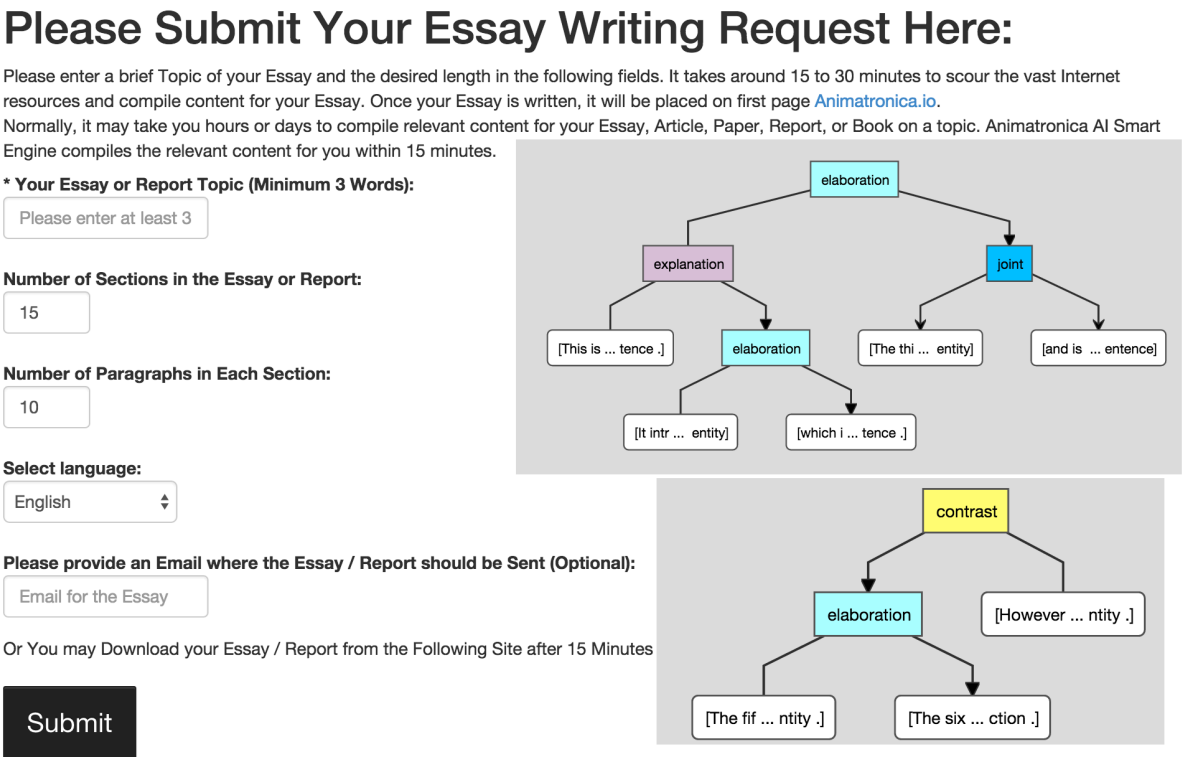

Figure 1: Content Compilation front end (on the left). The pair of discourse trees to find an appropriate sequence of mined text fragments (on the right-bottom)

2 Text Fragment Mining Algorithm

To write a document, we first create its table of contents (TOC). To do that, we mine the web for most important attributes associated with an entity we are writing about. For example, if we write a biography about a person, we find a biography page about a person of a similar kind (such as a writer or a scientist) and extract a TOC from it. Another option is two mine auto-complete values for this entity. For a scientist, it would be {*born, educated, researched, discovered, announced, became well known*}. Usually, Wikipedia is a good source of a structure of a TOC for a document on a given topic. TOC items will constitute a seed from which web search query will be formed.

The chart for text fragment mining algorithm is shown in Fig. 2. We start with the seed, one or multiple sentences each of which will form one or more paragraphs about the respective topics of the TOC. These seed sentences can be viewed as either headers or informational centroids of content to be compiled. We now iterate through each original sentence, build block of content for each and then merge all blocks, preceded by their seed sentences together, similar to (Sauper & Barzilay 2000).

To find relevant sentences on the web for a seed sentence, we form query as extracted significant noun phrases from this seed sentence: either longer one (three or more keywords, which means two or more modifiers for a noun, or an entity, such as a proper noun). If such queries do not deliver significant number of relevant sentences formed from search results, we use the whole sentence as a search engine query, filtering our content that is a duplicate to the seed.

The formed queries are run via search engine API or scraped, using Bing; search results are collected. We then loop through the parts of the snippets to see which sentences are relevant to the seed one and which are not. For all sentences obtained from snippets, we verify appropriateness to form content on one hand, and relevance to the seed sentence on the other hand. Appropriateness is determined based on grammar rules: to enter a paragraph cohesively, a sentence needs to include a verb phrase and be opinionated (Galitsky et al 2009). We filter out sentences that look like one or another form of advertisement, a call to buy a product, or encourages other user activity by means of an imperative verb.

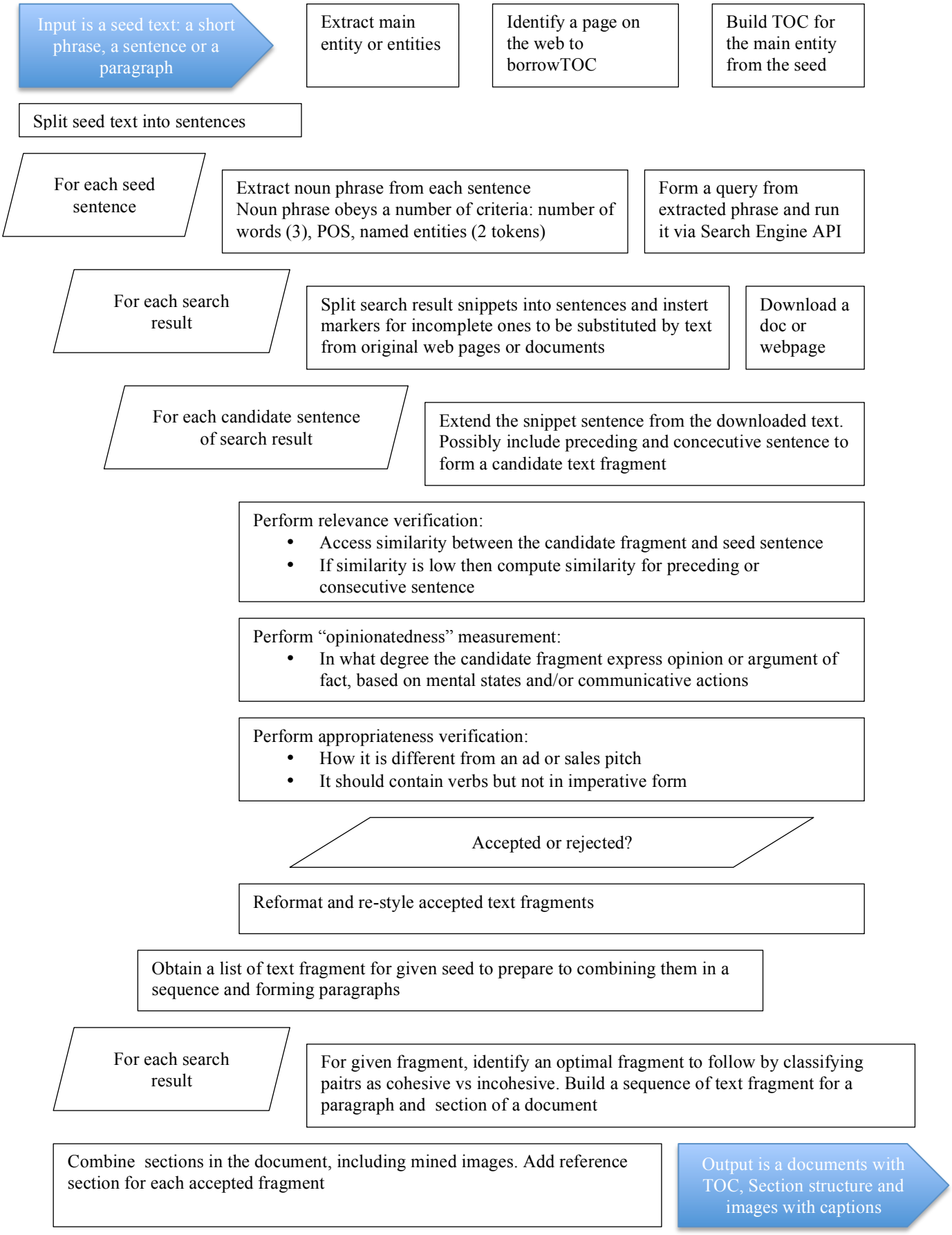

Figure 2: Content compilation algorithm

Relevance is determined based on the operation of syntactic generalization (Galitsky et al 2012), where the bag-of-words approach is extended towards extracting commonalities between the syntactic parse trees of seed sentence and the text mined on the web. Syntactic generalization score is computed as a cardinality of maximal common sub-graph between the parse trees of the seed and candidate sentences or text fragments. Syntactic generalization allows a domain-independent semantic measure of topical similarity, delivering stronger relevance than the search engine itself or the keyword statistics.

In addition to syntactic generalization, the tool verifies common entities between seed and mined sentence, and applies general appropriateness metric. The overall score includes syntactic generalization score (the cardinality of maximal common system of syntactic sub-trees) and appropriateness score to filter out less suitable sentences. Finally, mined sentences are re-styled and re-formatted to better fit together. The following section explains how paragraphs are formed from text fragments.

3 Arranging Candidate Text Fragments

To form a coherent sections of a document, text fragments need to agree. For a given candidate fragment, we either find its optimal position in a section of a document for the receding and following fragment or paragraph of text, or reject it. To implement this functionality, we build a classifier for a pair of consecutive text fragments (paragraphs) and classify them as a valid (coherent, acceptable agreement) pair or an invalid one (Galitsky et al., 2015). We use a discourse trees representation (Joty et al 2013) where the parse tree information for each elementary discourse unit is retained. To form <*Fragment1, Fragment2*> pair one can combine the respective discourse trees into a single tree with the root RR (Fig.3). The discourse trees for these pairs are subject to tree kernel learning (Zhang & Lee 2003). We form a positive training set of classifier from the pairs of paragraph which actually follow each other and a negative training set - from the ones randomly selected from text (Yahoo! Answer corpus was used).

4 Conclusions

The discourse tree representation used in our content compilation system is a reduction of what is called parse thicket (Galitsky et al., 2015), a combination of parse trees for sentences with discourse-level relationships between words and parts of the sentence in one graph. The straight edges of this graph are syntactic relations, and curvy arcs – discourse relations, such as anaphora, same entity, sub-entity, rhetoric relation and communicative actions. This graph includes much richer information than just a combination of parse trees for individual sentences would. Parse thickets can be generalized at the level of words, relations, phrases and sentences (Fig. 3).

The tool has been advertised using Google AdWords and used by thousand of users searching for "free essay writing" to compile content for a variety of domains, including natural sciences and humanities.

The system is available for general audience at http://animatronica.io/submit.html. Examples of written documents on a wide variety of topics is available at http://mail3.fvds.ru/wrt_latest/.The source code can be obtained at https://github.com/bgalitsky/relevance-based-on-parse-trees under Apache Licence and is a sub-project of Apache OpenNLP https://opennlp.apache.org/.

Reference

Liapis, Antonios, Georgios N Yannakakis,and Julian Togelius. 2013. Sentient Sketchbook: Computer-aided game level authoring." InFDG, 213–220.

Johnson, MR, 2016. Procedural Generation of Linguistics, Dialects, Naming Conventions and Spoken Sentences. Proceedings of 1st International Joint Conference of DiGRA and FDG.

Galitsky, B., Ilvovsky, D., Kuznetsov, S. O. 2015. Text Classification into Abstract Classes Based on Discourse Structure. Proceedings of Recent Advances in Natural Language Processing, pages 200–207, Hissar, Bulgaria, Sep 7–9 2015.

Galitsky, B., MP González, CI Chesñevar.. A novel approach for classifying customer complaints through graphs similarities in argumentative dialogue. *Decision Support Systems*, Volume 46, Issue 3 717-729.

Galitsky, B., Gabor Dobrocsi, Josep Lluis de la Rosa, 2012. Inferring the semantic properties of sentences by mining syntactic parse trees. Data & Knowledge Engineering v81 pp 21-45.

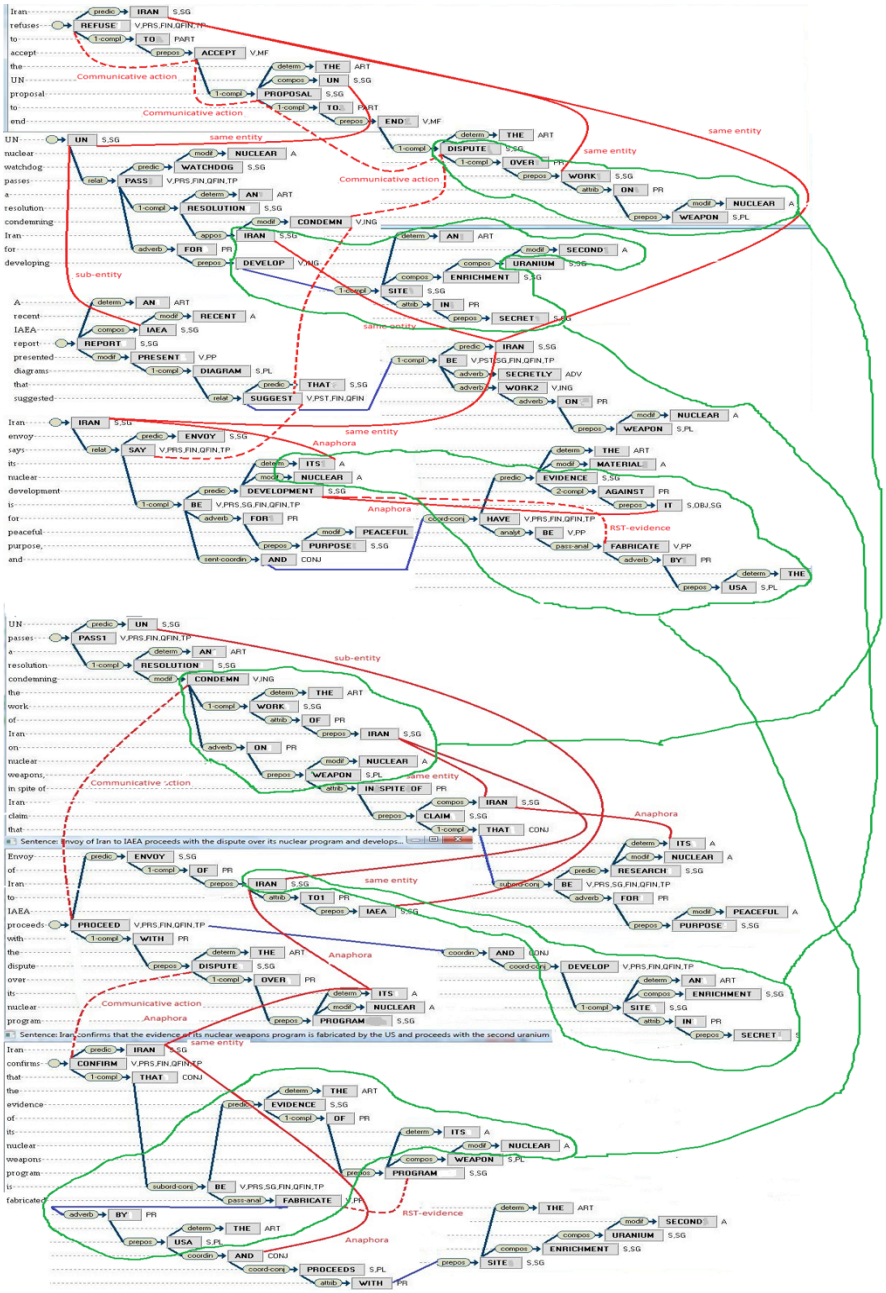

Figure 3: Parse thickets of two paragraphs assuring document cohesiveness

Hendrikx, Mark, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. 2013. Procedural content generation for games: A survey. ACM Trans. Multimedia Comput. Commun. Appl. 9, 1, Article 1 22 pages.

Zhang, Dell and Wee Sun Lee. 2003. Question classification using support vector machines. In SIGIR '03: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 26–32, New York, NY, USA, ACM.

Galitsky, B. 2013. Machine Learning of Syntactic Parse Trees for Search and Classification of Text. Engineering Application of Artificial Intelligence, dx.doi.org/10.1016/j.engappai.2012.09.017.

Joty, Shafiq R, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. Combining intra-and multi- sentential rhetorical parsing for document-level dis- course analysis. In *ACL (1)*, pages 486–496.

Sauper, Cristina and Regina Barzilay. 2000. Automatically Generating Wikipedia Articles: A Structure-Aware Approach, Proceedings of ACL.