A Modular Tool for Automatic Summarization

Valentin Nyzam, Aurélien Bossard

A Modular Tool for Automatic Summarization

Valentin Nyzam
LIASD - Université Paris 8
v.nyzam@iut.univ-paris8.fr

Aurélien Bossard
LIASD - Université Paris 8
a.bossard@iut.univ-paris8.fr

Abstract

This paper introduces the first fine-grained modular tool for automatic summarization. Open source and written in Java, it is designed to be as straightforward as possible for end-users. Its modular architecture is meant to ease its maintenance and the development and integration of new modules. We hope that it will ease the work of researchers in automatic summarization by providing a reliable baseline for future works as well as an easy way to evaluate methods on different corpora.

1 Introduction

Automatic summarization (AS) is studied since the late 1950s (Luhn, 1958). Automatic summarization methods were mostly extractive until recently, where abstractive methods have emerged thanks to the recent breakthrough in neural networks. Abstractive automatic summarization methods are for the most part supervized. However, because of the automatic summarization task complexity, huge corpora made of pairs of documents and their associated summary are needed. For example, the CNN and Dailymail news corpora on which are based the first neural-based news automatic summarizers are composed of more than 200,000 pairs of document and summary. Other summarization tasks can also be provided with large corpora, such as scientific articles summarization. However, most of real-life summarization tasks come without any learning corpus. The cost in human resources to build such corpora is such that unsupervized summarization methods cannot be excluded from the research field.

In the last decades, efforts have been made in different fields of computer science to release open source systems that encode several methods. GATE[1](Dowman et al., 2005) platform is an example of such an open source system for NLP. Other research fields such as machine learning benefit from several open source modular systems, e.g., Weka (Hall et al., 2009), SPMF (Fournier-Viger et al., 2014), or Orange (Demšar et al., 2013). To our knowledge, no such tool exists for AS. A modular and open source tool for automatic summarization could allow to easily test different automatic summarization methods on different tasks / corpora. If such a tool is proven to be reliable, it can also be used as an acknowledged baseline for new systems – abstractive or extractive – to compare to. In fact, we found that some recent papers in neural abstractive summarization compare their results with naive extractive baselines or even no extractive baseline at all (e.g. (See et al., 2017) with a lead-based extractive baseline, (Chopra et al., 2016) with no extractive baseline at all). These works could have sorely benefited from a straightforward and easy-to-use summarization platform to establish fair comparisons to older extractive systems.

In this paper, we present an open source modular tool dedicated to automatic summarization. Written in Java, it is designed to first answer the lack of such a tool and so provide the community with an easy-to-use summarization tool, to allow a straightforward maintenance of existing modules and development of new modules, and to allow methods comparison in a unified framework. The tool is available on GitHub : https://github.com/ToolAutomaticSum/MOTS. We also present a study on DUC, TAC, CNN and Dailymail corpora.

[1] http://gate.ac.uk

2 Related Work

Open source summarizers can be classified into two categories: systems that only implement one or more methods defined by their authors as results of research and systems conceived as a way to implement existing methods. In the first category, one can cite MEAD (Radev et al., 2004a) that originally implemented centroid-based extraction method (Radev et al., 2004b) and later implemented LexRank (Erkan and Radev, 2004). However, this system does not seem to be available anymore. Among the other systems in that category, one can cite ICSISUMM (Gillick et al., 2009) that implements ILP-based summarization and MUSEEC (Litvak et al., 2016). However, systems in this first category cover only a few methods among existing methods, so there was a need for platforms with a better summarization methods coverage. SUMMA (Saggion, 2014) is a summarization toolkit implemented with GATE. It includes several sentence scorers, such as LexRank, Centroid and shallow features based. It benefits from GATE NLP methods. Sumy[2] is a more complete toolkit that implements eight different extraction methods, including baseline systems (random, first sentences only). As for PKUSumSum[3], (Zhang et al., 2016) implements ten different methods and handles three different summarization tasks: mono-document, multi-document and topic-based multi-document summarization. Modularity for these two system is however limited to tokenization/stemming and the choice of extraction method which is not decomposed itself in modules. This is the main asset of our tool: it is modular on a fine-grained level so automatic summarization methods are not defined globally but as a combination of small interchangeable modules. Table 1 shows details about the summarization methods implemented by SUMMA, Sumy and PKUSumSum.

3 Architecture

3.1 Modularity

Our tool is modular on a fine-grained level. Because of the modularity and the need of module compatibility definition, we made the choice of Java as programming language. Our tool can handle mono and multi-document summarization,

[2] https://github.com/miso-belica/sumy

[3] https://github.com/PKULCWM/PKUSUMSUM

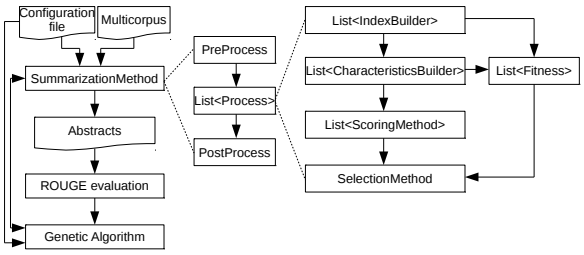

Figure 1: Architecture and workflow of our tool

topic-based or not. It embeds a genetic algorithm to tune hyperparameters. The summarization modules are all language-independent for multilingual summarization. Its architecture is conceived for both extractive, semi-extractive and abstractive paradigms. Also, using it as an end-user is straightforward.

It is divided in two branches: one for traditional greedy extractive methods, and one for global search algorithms such as genetic or knapsack algorithms (cf Figure 1). Our system can also handle fully abstractive methods.

The greedy branch is divided in four steps:

- IndexBuilder: an index is built using unigrams or n-grams for which a set of features is built;
- CharacteristicsBuilder: a set of features is computed for every text chunk[4] based on one or more indexes;
- SentenceScoring: score computation for every text chunk based on previous features;
- SelectionMethod: text chunks extraction with a method using previous scores.

The global search algorithms branch is divided in two steps:

- FitnessScore: a score computed for a candidate summary;
- SelectionMethod: the search algorithm itself guided by the fitness.

The four steps (or atomic processings) of a summarization method are independent, and they communicate via the Process class that controls their execution and compatibility. Input and output of the atomic processings are specified via inheritance of a specific method and the implementation of interfaces defined and documented in our tool. These interfaces make the Process class able to use Java methods to adapt input and output between each atomic processing. All atomic processings are independent and follow compatibility rules, so

[4] e.g., sentences, phrases, defined during preprocessing

our system architecture is completely modular.

3.2 Embedded evaluation

If the gold standard summaries are provided with the corpus to summarize, our tool can perform a call to ROUGE (Lin, 2004) in order to instantly retrieve the results of a summarization method on a specific corpus.

3.3 Embedded genetic algorithm

All summarization methods have parameters that influence the quality of the summaries. Optimizing these parameters for a specific task is crucial. Litvak et al. (2010); Bossard and Rodrigues (2011) have shown that a genetic algorithm (GA) can be efficient for this kind of optimization. Our tool integrates a GA to optimize summarization methods hyperparameters. We specified a *dna* syntax for the definition of methods hyperparameters that can be used by any module implemented in our tool. The GA uses ROUGE-2 as objective function, but it can be overrided using the modular architecture of our tool. The GA is launched using a XML configuration file that sets the hyperparameters to be optimized.

4 Implemented modules

The IndexBuilder class defines what the tokens are and how they are represented. We can use unigrams or n-grams and each of them can be associated with a frequency, a tf.idf value, or a vector representation computed with LSA or word embeddings.

The CharacteristicsBuilder class defines the representation of a text chunk (most of the time, text chunks are sentences). We can use a bag-of-words representation, the mean vector (the mean vector of all the tokens in a text chunk), the matrix composed of all tokens vectors, the co-occurrence graph (Rousseau and Vazirgiannis, 2013), the k-core representation (Batagelj and Zaversnik, 2003), or a clustering based on any representation (Bossard and Rodrigues, 2011).

The SentenceScore class defines how to compute a score for a text chunk depending on the characteristics computed previously. We can choose the sum of tf.idf above a threshold, the similarity with a vector or a matrix (to emulate (Radev et al., 2004b)), the position of the text chunk in a document.

The SelectionMethod class defines how the sentences are selected. We can chose greedy algorithms such as MMR (Carbonell and Goldstein, 1998), CSIS (Radev et al., 2004a), an extraction method based on a previous clustering (Bossard, 2013) or a naive extraction of the best sentences, or global search algorithms such as Knapsack (Gillick et al., 2008), a genetic algorithm (Bossard and Rodrigues, 2017), ILP (Gillick et al., 2009) or a reinforcement algorithm (Ryang and Abekawa, 2012).

Our tool can also call an abstractive external summarization method, retrieve the results and use them for postprocessing or evaluation purposes. This is not trivial as the index has to be updated in order to take into account out of vocabulary words.

Combining these modules, we can emulate most of the most known summarization methods. Table 1 shows a comparison between our tool and other summarization tools introduced in Section 2. It shows that, to our knowledge and at the moment we write this paper, our tool covers the most of the summarization methods covered by other known summarizers. Moreover, two of the three methods not yet implemented: Manifold rank (Wan et al., 2007) and Submodular functions (Lin and Bilmes, 2011) are currently under development and should be released soon.

5 Using our tool

Using our tool as an end user is straightforward. It only requires a configuration file that describes the summarization method to use by defining every module used and their parameters, and a descriptor file for the multicorpus to summarize. Even if a configuration file can be written from scratch, we supply standard configuration files that encode the most known and used summarization methods.

6 Study

We evaluated some summarization methods from our tool on different corpora: DUC 2006 and 2007 and TAC 2008, 2009 and 2010 corpora (multidocument news summarization) and CNN/Dailymail corpus.

We used 264.999 documents of the merged corpus of CNN and Dailymail to train the pointer-generator abstractive method. It was then validated on 11.659 documents and tested on 12.143 documents. We used the same evaluation set for all methods. We evaluated all methods with a limit

	Luhn	Edmundson	Lead	Centroid	LexRank	TextRank	KL incr.	Manifold Rank	Clust.	ILP	LDA	Submodular	Knapsack	Genetic	Pointer Generator
SUMMA	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Sumy	✓	✓	✗	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗
PKUSumSum	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✗	✗	✗
Our tool	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓

Table 1: Comparison of systems summarization methods handling capabilities

of 100 and 50 words, because we found out that Pointer-Generator often produces only 50 words long summaries.

As DUC and TAC corpora do not provide enough data to train a neural network, we used the model learned on CNN/Dailymail to generate a summary for DUC and TAC and applied it on the most recent newswire article of each set of documents.

Table 2 shows the results of a small sample of methods from our tool on DUC, TAC and CNN/DailyMail corpora. We made the choice of selecting methods that show the modularity of our tool:

- lead: the first n words of a document (for DUC and TAC the first n words of the most recent document);
- tf.idf MMR: sentences are scored with the sum of tf.idf scores and selected with MMR;
- Centroid MMR: sentences are scored with the Centroid method and selected with MMR;
- 2G Centroid MMR: sentences are scored with the Centroid method on bigrams and selected with MMR;
- 3G Centroid MMR: sentences are scored with the Centroid method on trigrams and selected with MMR;
- LexRank MMR: sentences are scored with the LexRank method on unigrams and selected with MMR;
- 2G LexRank MMR: sentences are scored with the LexRank method on bigrams and selected with MMR;
- 2G Centroid KS: sentences are scored with the LexRank method on bigrams and selected with a knapsack algorithm;
- 2G JS KS: sentences are extracted with a knapsack algorithm that uses the Jensen-Shannon divergence as fitness and bigrams as tokens;
- ILP: sentences are extracted with an ILP based solver under the constraints of (Gillick et al., 2009);
- Genetic: sentences are extracted with a genetic algorithm that uses the Jensen-Shannon divergence as fitness and bigrams as tokens;
- Pointer Generator: our tool calls the Pointer Generator (See et al., 2017) and retrieves its results.

DUC and TAC were evaluated using the best ROUGE parameters for these corpora in Graham (2015)'s study. CNN/Dailymail tasks were evaluated using a standard configuration of ROUGE: recall as score, bigrams as tokens, no stemming, removal of stop words.

As one can see in Table 2, 2G JS KS, ILP and Genetic methods perform badly on the CNN/DailyMail task. This is due to the fact that ILP method is designed for multidocument summarization. 2G JS KS and Genetic use the same fitness: a Jensen-Shannon divergence. We hypothesize that the CNN/DailyMail documents are too small for such a fitness based on the bigrams probability distribution. Without any suprise, Pointer Generator performs badly on DUC and TAC corpora. Even if the corpora are close (newswire articles for both DUC/TAC and CNN/DailyMail), the task is not exactly the same. This confirms that sometimes, unsupervized extractive methods are the only solution available, and that such methods shall not yet be laid aside by the research community.

7 Conclusion

This paper introduces a new tool for automatic summarization. Written in Java, it is completely modular and can emulate most of the most known extractive summarization methods. The tool is open source, and modules can be added easily. Compared to other existing tools, ours is modular on a fine-grained level, so a summarization method can be defined as a combination of different modules: token representation, text chunk representation, text chunk scoring, and text chunk se-

	DUC2006	DUC2007	TAC2008	TAC2009	TAC2010	CNN/DM(100)	CNN/DM(50)
lead	6.69	8.76	8.81	7.44	6.58	19.62	9.51
tf.idf MMR	8.08	9.48	7.72	7.1	7.72	19.71	10.69
Centroid MMR	8.73	9.34	8.39	7.63	7.93	21.15	12.55
2G Centroid MMR	9.88	11.21	10.84	9.29	10.15	21.58	11.85
3G Centroid MMR	9.10	10.94	10.84	9.29	10.40	21.14	11.50
LexRank MMR	8.83	10.44	8.98	9.36	9.24	21.17	10.96
2G LexRank MMR	8.37	9.82	8.62	8.9	8.94	17.62	10.78
2G Centroid KS	8.85	9.34	8.4	9.03	9.65	18.47	10.69
2G JS KS	10.18	12.61	11.28	11.35	10.01	14.92	8.55
ILP	9.63	11.35	10.96	9.88	10.43	14.89	5.44
Genetic	10.55	12.17	11.01	10.62	10.79	15.35	10.51
Pointer Generator	3.07	7.47	3.33	5.75	4.26	10.24	10.24

Table 2: Results on DUC, TAC and CNN/Dailymail corpora

lection. As we write this paper, and to our knowledge, our tool covers more automatic summarization methods than the three other existing summarizers.

As an end-user, using our tool, available on GitHub (`https://github.com/ToolAutomaticSum/MOTS`), is straightforward. It only needs a description of the corpus to summarize and a configuration file that describes the modules to use. We provide configuration files for the most known summarization methods.

For this paper, we evaluated a small sample of the methods that can be ran with our tool. Except for three methods that are really specific to multidocument summarization, the evaluated summarization methods beat the naive baseline that extracts the n first words from a document. This is still a competitive baseline when summarizing newswire articles.

Due to its ease of use and to its results on different summarization tasks, our tool can be used as a baseline for forthcoming research on automatic summarization.

Acknowledgement

This work is supported by a public grant overseen by the French National Research Agency (ANR) as part of the "Young researchers program" (reference : ANR-16-CE38-0008 ASADERA).

References

Vladimir Batagelj and Matjaz Zaversnik. 2003. An o(m) algorithm for cores decomposition of networks. *CoRR*, cs.DS/0310049.

Aurélien Bossard. 2013. Generating update summaries: Using an unsupervized clustering algorithm to cluster sentences. In *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 205–227. Springer.

Aurélien Bossard and Christophe Rodrigues. 2011. Combining a multi-document update summarization system–cbseas–with a genetic algorithm. In *Combinations of intelligent methods and applications*, pages 71–87. Springer.

Aurélien Bossard and Christophe Rodrigues. 2017. An evolutionary algorithm for automatic summarization. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 111–120, Varna, Bulgaria. INCOMA Ltd.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan. 2013. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353.

Mike Dowman, Valentin Tablan, Hamish Cunningham, and Borislav Popov. 2005. Web-assisted annotation, semantic indexing and search of television and radio news. In *Proceedings of the 14th International World Wide Web Conference*, Chiba, Japan.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of AIR*, 22:457–479.

P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C. Wu., and V. S. Tseng. 2014. SPMF: a Java Open-Source Pattern Mining Library. *Journal of Machine Learning Research (JMLR)*, 15:3389–3393.

Daniel Gillick, Benoit Favre, and Hakkani-Tür. 2008. The icsi summarization system at tac 2008. In *Proc. of the Text Analysis Conference workshop*.

Daniel Gillick, Benoit Favre, Dilek Hakkani-Tür, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The icsi/utd summarization system at tac 2009. In *Proc. of the Text Analysis Conference workshop, Gaithersburg, MD (USA)*.

Yvette Graham. 2015. Re-evaluating automatic summarization with bleu and 192 shades of rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Lisbon, Portugal. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics.

Marina Litvak, Mark Last, and Menahem Friedman. 2010. A new approach to improving multilingual summarization using a genetic algorithm. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 927–936. Association for Computational Linguistics.

Marina Litvak, Natalia Vanetik, Mark Last, and Elena Churkin. 2016. Museec: A multilingual text summarization tool. In *Proceedings of ACL-2016 System Demonstrations*, pages 73–78. Association for Computational Linguistics.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165.

Dragomir R Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, et al. 2004a. Mead-a platform for multidocument multilingual text summarization. In *LREC*.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004b. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40:919–938.

François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and tw-idf: new approach to ad hoc ir. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 59–68. ACM.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 256–265. Association for Computational Linguistics.

Horacio Saggion. 2014. Creating summarization systems with summa. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2903–2908, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jianmin Zhang, Tianming Wang, and Xiaojun Wan. 2016. PKUSUMSUM : A java platform for multilingual document summarization. In *COLING (Demos)*, pages 287–291. ACL.