

Generating Acrostics via Paraphrasing and Heuristic Search – Final Presentation

Bruno Soares Fillmann
Fernando Bombardelli da Silva
Jürgen Bauer
William Bombardelli da Silva

Technische Universität Berlin
Datenbanksysteme und Informationsmanagement
DBPRO – Database Projects (WS 2014/2015)

09.02.2015

Organization

- 1 Goals
- 2 Subtasks
- 3 Method
- 4 Difficulties
- 5 Unreached Goals
- 6 Example
- 7 Results
- 8 References

The Problem

- Given a text T and an acrostic X, find a paraphrased version of T that encodes X in the first letters of each line.

Original Text: ¹

Zuerst wollte er mit dem unteren Teil seines Körpers aus dem Bett hinauskommen, aber dieser untere Teil, den er übrigens noch nicht gesehen hatte und von dem er sich auch keine rechte Vorstellung machen konnte, erwies sich als zu schwer beweglich; es ging so langsam; und als er schließlich, fast wild geworden, mit gesammelter Kraft, ohne Rücksicht sich vorwärtsstieß, hatte er die Richtung falsch gewählt...

Result Text:

Zuerst wollte er mit dem unteren Teil seines Körpers a-
Us dem Bett hinauskommen, aber dieser untere Teil, den
Er übrigens noch nicht gesehen hatte und von dem er sich auch keine
Rechte Vorstellung machen konnte, erwies sich als zu
Schwer beweglich; es ging so langsam; und als er schließlich, fas-
t wild geworden, mit gesammelter Kraft, ohne Rücksicht sich
vorwärtsstieß, hatte er die Richtung falsch gewählt...

¹<http://www.gutenberg.org/browse/languages/de>

Goal of the Project

Goal: Implement the methods and techniques to generate acrostics for the German language as described in the paper *Generating Acrostics via Paraphrasing and Heuristic Search* of Benno Stein, Matthias Hagen, and Christof Bräutigam, published in August 2014 [1].

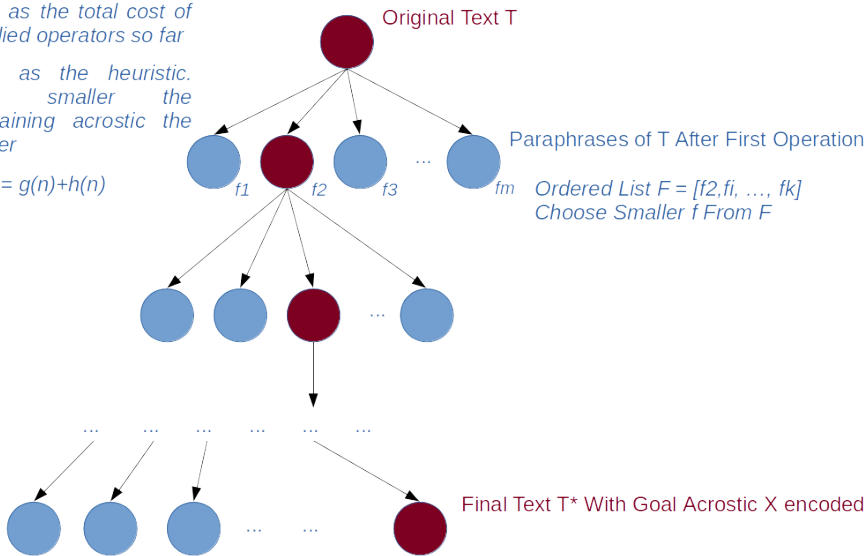
Main idea of the algorithm:

- Modeled as a search problem in a tree.
- The vertices are states (texts) and the edges are operations over states.
- Artificial intelligence is applied for the search strategy (A* Algorithm).

$g(n)$ as the total cost of applied operators so far

$h(n)$ as the heuristic.
The smaller the remaining acrostic the better

$$f(n) = g(n) + h(n)$$



Subtasks of the original goal

- Understand the paper and the method
- Design an architecture for the application
- Collect comparable German text corpora, statistics and libraries (TEX Hyphenation, NetSpeakAPI, Thesaurus, etc.)
- Implement the most promising paraphrasing operators (line break, hyphenation, wrong hyphenation, word insertion/deletion, synonym, spelling)
- Implement the A^* -algorithm
- Evaluate the results

How have we reached the goals?

- UML-diagram for the application
- Sequence diagram for the A^* -algorithm
- Developed the application in Java 8 with Netbeans
- Used various libraries, web services and databases to implement the operators
- Implemented a test application for evaluating the results, with a timeout after 15 min

Which difficulties did occur?

- Synonym operation only inserts synonyms without considering the context (Google N-Gram)
- Lack of powerful grammatical operator
- NetSpeakAPI doesn't yield many results for the German language
- NetSpeakAPI's requests to the web server takes much time
- The empirical choice of costs is not optimal for the search

Which difficulties did occur?

- Organization of the input text as line:
 - More than one letter of the acrostic can be generated at a time, not following the theoretical framework of the A*-algorithm as described in the paper.
 - **Recommendation:** view the text as one single line and apply line break only to generate a letter. When the acrostic is generated, break the lines according to the constraints.

Unreached Goals

- Only the most promising operators were implemented.
- No effective operator to generate the first letter of the acrostic.
- No operator **not** mentioned in the paper was coded.
- No local databases for *n-gram* were implemented.
- No sophisticated quality measures were implemented.













Example

Original Text: ²

Billig nach Hamburg. 24/7 taxidienst bietet ihnen kostengünstige und zuverlässige verkehr aus Kiel. Mit erfahrung in der branche wir garantieren die niedrigsten preis nach Hamburg. Gleiche preis gilt auch wenn sie taxi aus Kiel stadt nach Hamburg nehmen.

Final Text:

Billig nach Hamburg. 24/7 taxidienst bietet ihnen kostengünstige und zuverlässige verkehr aus Kiel. Mit erfahrung in der branche wir garantieren ihnen die niedrigsten preis nach Hamburg. Gleiche preis gilt auch wenn sie taxi aus Kiel stadt nach Hamburg nehmen.

²http://german.airportljubljana.co/Billig_nach_Klagenfurt.php            

Example

Original Text: ³

Diese war unterdessen in das Aussehen gerückt, was die Gegenwart damals hatte, sie war sehr schön, aber immer traurig, weil sie sich vor ihrem Bräutigam fürchtete und weil sie von den Schwestern, die keinen Mann bekommen, beständig seinetwegen geneckt wurde. Eines Tages rief ein heller Trompetenschall alle drei Schwestern ans...

Final Text:

Diese war unterdessen in das Aussehen gerückt, was die Gegenwart in jenen tagen hatte, sie war sehr schön, aber immer traurig, weil sie sich vor ihrem Bräutigam fürchtete und weil sie von den Schwestern, die keinen Mann bekommen, beständig seinetwegen geneckt wurde. Eines Tages rief ein heller Trompetenschall alle drei Schwestern ans...

³<http://www.gutenberg.org/browse/languages/de>

Results – Evaluation

- 100 Texts.
- 4 different ways.
 - A* + Self-referential:** Regular A* as the algorithm and the first word of the text as the acrostic.
 - A* + Most Common:** Regular A* as the algorithm and the most common word in German, that starts with the first letter of the text, as the acrostic.
 - Greedy A* + Self-referential:** Greedy A* (ignore costs of operations) as the algorithm and the first word of the text as the acrostic.
 - Greedy A* + Most Common:** Greedy A* as the algorithm and the most common word in German, that starts with the first letter of the text, as the acrostic.

Results

Configuration	Success Rate	Success Run-time	Nodes When Success	Timeout Rate	Nodes When Time-out
A* + Self-referential	37,62%	31,97 s	46112,82	62,38%	675794,78
A* + Most Common	64,36%	29,74 s	52547,68	35,64%	568845,14
Greedy A* + Self-referential	60,40%	30,64 s	24033,31	39,60%	488660,52
Greedy A* + Most Common	76,24%	9,40 s	24639,19	23,76%	304303,41

Table : Experimental results

Results – Frequency of the operators

Operation	A*	Greedy A*
LineBreak	23,74%	14,29%
WrongHyphenation	39,39%	31,63%
Hyphenation	5,05%	0,34%
WordInsertionDeletion	0,51%	0,34%
Synonym	29,80%	45,58%
WrongSpelling	1,52%	7,82%

Table : Operations employment frequency

References



Benno Stein, Matthias Hagen, and Christof Bräutigam. *Generating Acrostics via Paraphrasing and Heuristic Search*.

In Junichi Tsujii and Jan Hajic, editors, 25th International Conference on Computational Linguistics (COLING 14), pages 2018-2029, August 2014. Association for Computational Linguistics.



Martin Potthast, Martin Trenkmann, and Benno Stein. *Netspeak: Assisting Writers in Choosing Words*.

In Cathal Gurrin et al, editors, Advances in Information Retrieval. 32nd European Conference on Information Retrieval (ECIR 10) volume 5993 of Lecture Notes in Computer Science, pages 672, Berlin Heidelberg New York, March 2010. Springer.



Private Communication with Rainer Perkuhn, Institut für Deutsche Sprache Programmbereich Korpuslinguistik, via Email, in Dezember 2014. He send us two files of an old study of Cyril Belica, including the first letter frequencies.



Dechter, Rina; Pearl, Judea. *Generalized best-first search strategies and the optimality of A**. Journal of the ACM (JACM), Volume 32 Issue 3, Pages 505-536, New York, July 1985.

Questions?