Using GermaNet for the Generation of Crossword Puzzles

Claus Zinn and **Marie Hinrichs** and **Erhard Hinrichs**
Department of General and Computational Linguistics
University of Tübingen
Keplerstraße 2
72074 Tübingen, Deutschland
Correspondence: claus.zinn@uni-tuebingen.de

Abstract

Wordnets are playing an important role in research, but so far they have found little use in practical applications that are aimed at the general public. In this paper, we present a crossword generator that exploits lexical-semantic resources such as GermaNet. The software is capable of (i) automatically filling in the grid of a crossword puzzle with words taken from GermaNet for variable grid sizes, and (ii) generating clues for each word that is included in the grid. Crossword generation is not trivial, and we report on the effectiveness of various heuristic search functions that we have used.

1 Introduction

Crossword puzzles play with words. A puzzle is usually presented as a rectangular grid of black and white squares. The game's objective is to fill the white squares with letters, forming words that intersect with each other. Words, and their letters, can be written horizontally and vertically. Black squares serve as separators between words. Words are not arbitrary. For each word, there is a textual clue that describes it.

The New York Times (NYT) is well-known for its daily crosswords, and it even offers a site where useful information about its puzzles is published.[1] According to the site, the NYT uses a variety of clue types such as puns, anagrams, cryptic clues and even sound clues. The clues describe words that cover a variety of different topics, *e.g.*, television shows, movies, classical music, art, and history. Moreover, the Sunday puzzles have a theme, which is referenced in a humorous quotation or pun found in the answers. Also, Friday/Saturday puzzles tend to use longer words and are perceived as more complex than the puzzles for the other week-days. Fig. 1 describes a puzzle with a 5×5 grid taken from (Ginsberg et al., 1990). The puzzle,

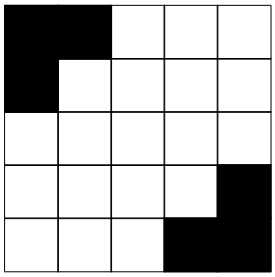

Figure 1: Example Puzzle.

with clues omitted, looks for five words each in *across* and *down* direction. Note that each word intersects with at least three other words so that they need to share the respective characters. Once all word slots are filled, clues must be generated that elicit each of the words, preferably, using interesting clues of different types.

The generation of crossword puzzles requires dictionaries and other lexical resources. In the past, an abundance of digital lexical resources have been created, for instance, GermaNet, the largest lexical-semantic word net for German (Hamp and Feldweg, 1997). It has to be said, however, that digital lexical resources are mostly used by researchers rather than the general public. We would like to boost usage of GermaNet by the general public in part by offering on-line access to popular games like crossword puzzles and by developing software for generating such crossword puzzles automatically.

To attract the general audience to linguistic resources, we found crossword puzzles particularly intriguing.[2] In this paper, we report on our research using GermaNet to automatically solve crosswords puzzles such as the ones given in Fig. 1. With Ger-

[1] https://www.nytimes.com/article/how-to-solve-a-crossword-puzzle.html

[2] As their everyday occurrences in newspapers testify, crossword puzzles are very popular. In Germany, for instance, 56 from 100 persons do a crossword puzzle at least once a year; 40% do a puzzle at least once a month, and 21% do a crossword once a week, see https://www.freizeitmonitor.de/2023/alle-freizeitaktivitaeten-im-ueberblick/.

maNet holding over 215,000 lexical units, there is an abundance of choice points an algorithm must take into consideration. As a result, the branching factor of the resulting search tree is rather large, and to conquer it, heuristic information is required to solve non-trivial crossword puzzles. Once lexical entries have been assigned to word slots, the identification or generation of clues to hint at them – in an overall entertaining manner – is also harder than thought.

The remainder of the paper is structured as follows. Sect. 2 gives an overview of GermaNet, with a particular focus on using this resource for solving crossword puzzles. It also reviews some of the literature on crossword generation. In the main part of the paper, we discuss our algorithm for crossword generation using GermaNet (Sect. 3), which is followed by an evaluation. In Sect. 4, we give a brief overview on clue generation. A front-end GUI is presented in Sect. 5, and Sect. 6 discusses our work, future work, and concludes.

2 Background

2.1 GermaNet

GermaNet is the largest lexical-semantic word net for German (Hamp and Feldweg, 1997). The development of the resource started 25 years ago, and is still actively maintained and enriched.[3] The latest version of GermaNet (18.0) features 215,000 lexical units that are attached to 167,163 synsets. It has 181,530 conceptual relations, and 12,602 lexical relations (synonymy excluded). Furthermore, GermaNet has a representation of 121,655 split compounds, and it includes 28,563 pointers into the Interlingual Index. Moreover, GermaNet has 11,760 paraphrases attached to synsets. Also, 29,550 sense definitions were added from Wiktionary in 2011[4] (Henrich et al., 2014). A clue in a crossword is always tied to a word slot of a given length. Fig. 2 depicts the distribution of GermaNet lexical entries in terms of word length. Words longer than 25 characters are omitted.[5] It shows that GermaNet's database also covers the short and long word spectrum very well.

Tab. 1 depicts the potential of using GermaNet for the generation of crossword puzzle clues. In addition to the use of 11,7k paraphrases and the 29.5k sense descriptions to generate definitional clues, we also exploit relationships between lexical entries and between synsets. For now, we limited ourselves to only use two conceptual and one lexical relation to construct three other types of clues, namely, hypernyms (using 171,925 relation instances), synonyms (lexical units being in the same synset, 143,534), and antonyms (3,982).

It should be noted, however, that the generation of clues that ask for synonyms need special care. Consider, for instance, the use of synonyms in the thematic domain 'human'. Here, a synset usually contains both the male and female form. For example, all of the four lexical units 'Dermatologin', 'Hautärztin', 'Hautarzt', 'Dermatologe' (engl.: dermatologist) are part of the same GermaNet synset. It would provide little entertainment to search for the word 'Hautärztin' with the clue 'Synonym für Hautarzt'. However, searching for the word 'Dermatologe' is much more appropriate in a crossword setting. To avoid the generation of trivial clues, we only use two synonyms when there is little string overlap between them.

Clearly, the paraphrase and wiktionary information provide the most verbose clue to a given word. From our own experience, those clues are refreshingly new when compared to often repeated or well-known clues that one encounters in crosswords in newspapers and puzzle books. Given the aforementioned constraints, with the combination of paraphrases, wiktionary entries, synonyms, hypernyms, and antonyms, together with the future use of other relations (*e.g.*, meronyms), the crossword generator can tap into a potential of 500k+ clue constructions for GermaNet-based puzzles.

2.2 Crossword Puzzle Generation

The generation and solving of crossword puzzles has been studied before. (Berghel, 1987) organises the problem into six distinct operations:

1. creation of the host matrix
2. determination of the overall design (i.e., pattern of open and closed cells) within the matrix

[3] The latest version was released in May 2023; for information to get access to the resource, see https://uni-tuebingen.de/en/142806.

[4] The entries were automatically mapped to lexical units in GermaNet and subsequently manually verified. In some cases, slight modifications to the Wiktionary sense descriptions have been made.

[5] For completeness: there are 797 words of length 26, 469 words of length 27, 243 words of length 28, 145 words of length 29, and 57 words of length 30. The longest word is *Finanzdienstleistungsaufsichtsbehörde* (engl. Financial Services Supervisory Authority) with 38 characters.

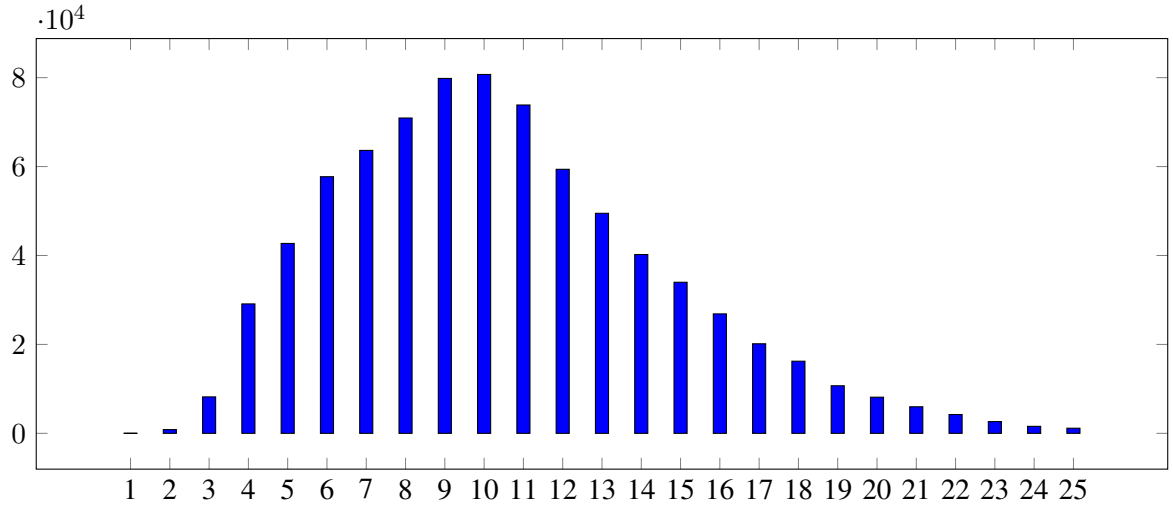

Figure 2: Word Length Distribution.

clue type	attached to	example clues	word slot value(s)
paraphrase	synset	ein gewisses Talent/eine bestimmte Begabung besitzend	talentiert, begabt
wiktionary	lexUnit	Medizin: den Gehörsinn oder das Gehörorgan betreffend; das Hören betreffend	auditiv
synonym	synset	Synonym für "Schlagbaum"	Schranke
hypernym	synset	Überbegriff für "Parietallappen"	Gehirnareal, Hirnareal, Hirnregion, Gehirnregion, Gehirnbereich, Hirnteil, Gehirnteil
antonym	lexUnit	Antonym für "konkret"	abstrakt

Table 1: Crossword Clues in GermaNet.

3. specification of word slots

4. identification of shared cells

5. construction of one or more solution sets, and

6. composition of a clue set for each solution set.

Berghel advocates a Prolog-based approach to solving crosswords, emphasising the declarative aspect of Horn Logic and how it allows stating the problem in a straightforward manner; a word in represented as a sequence of cells, and cells are represented as Prolog variables. When two words intersect, the respective cell shares the same Prolog variable. In a follow-up work, (Berghel and Yi, 1989) propose a procedure, *crossword compiler-compilation*, which will create source code for a crossword solver from the puzzle geometry alone.

It shows that the creation of Prolog code to solve crossword puzzles is rather straightforward. Consider, for instance, the following grid, a fully *interlocked* puzzle, *i.e.*, a puzzle with no black cells:

B	C	D
F	I	J
G	L	N

Here, each cell is assigned its own Prolog variable.

Now, assume lexical entries, and the clues that hint at them, being represented as Prolog-based `word/4` facts. Then, a straightforward implementation of blind, depth-first search can be implemented by the following Prolog program, with A, E, H, K, M, and O denoting words of length 3, and B, C, D, F,I, J, G, L, and N denoting the words' characters:

```
word(3, A, [B,C,D], C1), \+ member(A, []),
word(3, E, [B,F,G], C2), \+ member(E, [A]),
word(3, H, [F,I,J], C3), \+ member(H, [E, A]),
word(3, K, [C,I,L], C4), \+ member(K, [H, E, A]),
word(3, M, [G,L,N], C5), \+ member(M, [K, H, E, A]),
word(3, O, [D,J,N], C6), \+ member(O, [M, K, H, E, A]),
```

The C_i denote the clues to elicit the words. Note that words that intersect which each other share a letter such as the Prolog variable B; it is shared by the two words originating from the top-left corner in across and down direction. The `member/2` predicates ensure that no word is used twice.

Note that such Program code can be automatically generated for any given grid, and we have written such a meta-program. The programs it generates establish the base case for our evaluation.

In the remainder of this paper, we will focus on *given* puzzle grids, that is, predefined $x \times y$ matrices, potentially including black cells to add additional word boundaries. We consider Berghel's step 1-4 trivial and focus on step 5 and step 6.

The search space to conquer to fill all word slots is huge, and Breghel discusses some heuristics to guide this search. Heuristic information and their effectiveness have also been discussed by (Ginsberg et al., 1990), classifying four distinct types of choices that a puzzle solver must make:

1. which word slot to work on next?
2. which word should be used to instantiate the selected slot?
3. how to handle backtracking in cases where word slots become uninstantiable?
4. which kind of preprocessing is required?

(Smith and Steen, 1981), (Ginsberg et al., 1990) and (Ginsberg, 2011) all agree that the *hardest* slots should be considered next; these are the slots with the fewest alternatives, that is, the least number of possible instantiations with words. And since all slots must eventually be instantiated, the failure to instantiate the hardest one will initiate backtracking to undo former choices (see point 3 above).

Once a slot has been selected to work on, it should be instantiated with a word that restricts the possible choices for subsequent slots as little as possible (Ginsberg et al., 1990). Words with frequent letters will hence be preferred to words with less frequent ones. The computation of this heuristics is expensive so that only the value of the first k instantiations will be computed.

3 Solving crosswords with GermaNet

In this section, we give further details to apply the aforementioned heuristics for GermaNet.

3.1 Preprocessing

Given the RDF-based variant of GermaNet (Zinn et al., 2022), we have extracted relevant information via SPARQL queries and represent it as a list of `word/5` predicates, *e.g.*,

```
word(14, 1.0860799758322117, 'unregelmäßig',
 [u, n, r, e, g, e, l, m, a, e, s, s, i, g],
 [ literal('in zeitlich ungleichen Abständen
           wiederkehrend') ...]).
```

The first parameter gives the length of the word, the second parameter encodes a simple unigram frequency model, where the relative frequencies of a character with regard to the GermaNet lexicon are added up.[6] The solution word is given as third argument of `word/5`, whereas the fourth spells out the word; here, any German *Umlaut* is replaced with its corresponding two letters (*e.g.* ö $\rightarrow$ *oe*, or ß$\rightarrow$ *ss*). The last parameter of `word/6` gives the actual clues (only one clue is shown).

For the results reported in this paper, we have built two databases (one for unigram rankings, one of bigram rankings) for all GermaNet entries up to length 16. In total, 155k database entries have been constructed. Also, we have built a database of randomly-ordered word entries.

3.2 Heuristics

Our algorithm aims at replicating and finetuning the aforementioned heuristics for GermaNet. We hence follow a two-step approach. First, the hardest word slot is selected. Then, a word needs to be chosen to fit this slot. Such a word must maximise the satisfiability of the remaining open word slots. Both steps require word slots to be ranked.

Ranking of word slots. An *open* word slot of a given length L has exactly L variables, some of which may already be instantiated to characters; these are the cells that intersect with words already placed. A word slot is evaluated in terms of the number of words that can be placed into the slot. We give an example: the word slot [C1, C2, C3, C4], with all C_i being variables, is assigned the value $29,109$ because there are $29,109$ words of length four is GermaNet.; the word slot [e, C2, C3,e] has the value 21, because there are 21 words that fit the pattern (such as "Ente", "Este", "Eile", and "Ende").

Ranking of words to fit a given slot. Once the algorithm decided on a slot to work on next, a word has to be found to fit the slot. All word candidates are computed, and the one that maximises the satisfiability of all remaining open word slots is chosen. In line with (Ginsberg et al., 1990), we have introduced a k value which is used as follows:

[6] Similarly, lexical entries have been compiled with a bigram model.

Grid	# Slots	random			unigram			bigram		
		$k=1$	$k=5$	$k=10$	$k=1$	$k=5$	$k=10$	$k=1$	$k=5$	$k=10$
I 3x3	6	0.27	0.24	0.22	0.30	0.42	0.26	0.68	0.25	0.69
I 4x4	8	0.47	0.38	0.4	0.60	0.50	0.68	0.51	0.89	0.45
I 5x5	10	36.94	33.80	22.11	63.01	21.25	35.93	35.74	32.75	19.21
I 6x6	12	–	–	–	–	–	–	–	–	–
G 5x5	10	3.64	1.30	3.32	1.76	3.63	1.73	3.35	3.35	3.41
G 9x9	24	2.83	3.01	2.12	3.93	2.42	3.74	2.92	4.57	2.86
G 13x13 (a)	64	29.92	14.68	19.90	23.35	21.71	16.59	34.83	15.40	22.88
G 13x13 (b)	60	–	465.05	694.58	–	538.10	627.43	–	493.37	713.49

Table 2: Main algorithm using random word order, unigrams and bigrams – all decimals denote timings in seconds.

find all candidate words that fit a slot; rank them all, and then select the best k as word candidates. Only these k best candidates will be tried through backtracking.

Note that all `word/5` predicates are sorted via their respective n-gram value, that is, words with more frequent characters or bigrams are seen by Prolog first. As said, there is also a random ordering of such facts.

3.3 Evaluation

In this section, we evaluate the system in terms of the various heuristics employed to guide search. The base line is defined by blind-search using automatically generated Prolog programs from given grids (see Sect. 2).

Baseline Algorithm The Prolog programs were generated to alternate between filling *across* and *down* word slots. For this purpose, the order of the `word/5` were arranged accordingly.[7]

The following table depicts our baseline timings[8] with random, unigram and bigram ranking of the lexical entries:

Grid	# Slots	random	unigram	bigram
I 3x3	6	0.02	0.02	0.0.2
I 4x4	8	0.20	0.10	0.05
I 5x5	10	180.34	17.16	269.62
I 6x6	12	–	–	–
G 5x5	10	3.90	42.38	48.65
G 9x9	24	–	–	–
G 13x13 (a)	64	–	–	–
G 13x13 (b)	60	–	–	–

The first four test cases are fully interlocked grids; all remaining test cases are from (Ginsberg et al., 1990). It shows that the base program can solve fully interlocked puzzles up to grid size $5x5$, but fails to come up with a solution for larger ones (program stopped after 1 hour). The random word order performs surpringly well. In fact, the numbers indicate that the ordering of `word/5` facts in the Prolog database does not have a large impact, and that any outliers can be explained by having the right words in the right place by pure chance.

Main Algorithm. We evaluate the heuristic search algorithm using the same three conditions (random, unigram, bigram). Tab. 2 displays the main findings. It shows that the heuristics-driven algorithm pays off for crossword puzzles of larger grid sizes. For each condition, the same puzzles can be solved in less than 20 minutes; independent of the condition, the algorithm fails to solve the fully interlocked 6x6 grid as well as the 13x13 (b) puzzle with $k = 1$ in the threshold time.

Results of a linear mixed-effects regression model on cpu time (log-transformed) showed no interaction between k and type of model ($p > 0.9$).

However, there was a significant main effect of model. Pair-wise comparison showed that random is significantly faster than bigram ($p < 0.05$), while no other comparisons are significant.

Numerically, it seems that for more complex puzzles, a low k-value leads to longer processing times, but no significant differences can be found for less complex puzzles. Also, there seems to be little difference between using $k = 5$ and $k = 10$. Here, more test cases are required to determine whether the interaction between k and puzzle complexity is significant.

4 Clue Generation

Once the puzzle grid has been solved, with all words placed, clues must be generated to elicit them. With GermaNet having 11,760 paraphrases attached to synsets, and 29,550 sense descriptions attached to lexical units, the majority of GermaNet it "clueless" as it comes without this information. A

[7] An algorithm that does not alternate between across and down directions is significantly less efficient than one that does. The non-alternating algoritm is set to solving the crossword puzzle row by row, only to find out that "words" in down directions cannot be found in the lexicon. Here, the backtracking process is all but optimised.

[8] Results obtained by running SWI-Prolog on a recent Macbook Pro. All timings given in seconds.

Waagerecht 1 Großes Gewässer, das von Land umgeben ist. 3 Hundert Teile eines Euros. 5 Knapper Slip. 7 Möbelstück in der Küche zur Zubereitung von Mahlzeiten. 9 Bund fürs Leben. **Senkrecht** 2 Heißes Getränk, das aus getrockneten Blättern hergestellt wird. 4 Bargeld in physischer Form. 6 Werkzeug zur Wahrnehmung von Gerüchen. 8 Beengte Platzverhältnisse 10 Antonym für Anfang

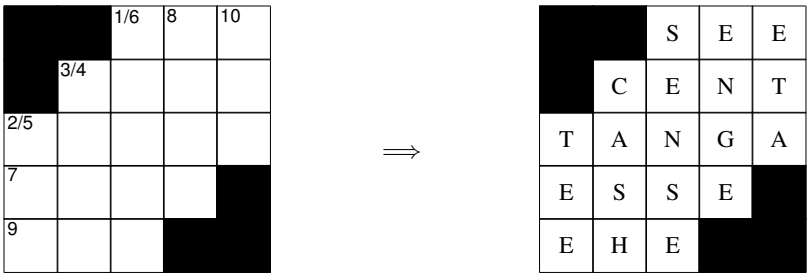

Figure 3: One possible solution to a given crossword grid.

large number of clues can be generated from hypernym and hyponym relations between synsets (*e.g.*, Überbegriff für "Nuss"), and antonym relations between lexical units (*e.g.*, Antonym für "lebendig") but the resulting puzzle would have little entertainment value if many clues for a puzzle were of this nature. To make clues more interesting, two steps were taken.

First, we ensure that the generation of clues observe a given distribution over their types. Here, we reserve at least 70% of clues to be paraphrases or sense descriptions; the remaining number of clues is evenly distributed over the other clue types (synonyms, hypernyms, and antonyms).

In the case of lexical units that are not paired with paraphrases or sense descriptions, we are using the ChatGPT *gpt-3.5-turbo* model for automatically generating paraphrases or sense descriptions. This process is fully automated. Reconsider the example puzzle given in the introduction. Fig. 3 depicts one of the possible solutions for the given grid. With the words being identified by our crossword solver, we asked ChatGPT (*gpt-3.5-turbo* model) to generates clues for each of the words using all possible clues types. For this purpose, we assigned the LLM the following assistant role:

> "You are generating clues for a German crossword puzzle. For the next word, generate a clue that describes the word, but which does not use any form of the word in the clue. The clues do not need to be full sentences, and should be as short as possible."

In this context, prompts specific to the clue type were asked, *e.g.*, "Schreibe in einem Satz einen Lexikoneintrag für: Tee" (engl. "Write, in one sentence, a lexicon entry for: tea"). The clues depicted in the top part of Fig. 3 show that ChatGPT is surprisingly good at generating crossword puzzle clues.

We are also looking forward to *officially* include resources of *Digitale Wörterbuch der deutschen Sprache*.[9] Consider, for instance, the word "Schranke" (engl.: barrier) for which we have generated the clue "Synonym für 'Schlagbaum'" (because both lexical units are in the same GermaNet synset). In the DWDS, the entry "Schranke" has been given the meaning "große, waagerecht oder senkrecht bewegbare Stange oder Gatter zur Absperrung von Durchgängen, Übergängen"[10], and the entry "Schlagbaum" has the meaning "Schranke, besonders an einer Grenze"[11], which are both good clue alternatives.[12]

5 GUI Interfaces

There is a significant difference between solving a given puzzle grid (as discussed so far), and the simultaneous process of generating and solving a grid, where new words entered in the grid can change the grid's layout, say, by adding new word boundaries (black cells). The latter task is much less constrained, and hence, much easier to tackle. In the past, we have implemented this easier task; we have also built a browser-based front-end as well as a LaTeX-based puzzle export function. This

[9] https://www.dwds.de

[10] https://www.dwds.de/wb/Schranke

[11] https://www.dwds.de/wb/Schlagbaum

[12] There are 78,815 entries in GermaNet without clue candidates (using hypernymy, hyponymy, and antonymy). For 11,439 of these entries, a paraphrase from DWDS can be found.

Figure 4: Screenshot of the web-based front-end.

front-end can be also used for our new algorithm presented herein.

5.1 Browser-based front-end

Our graphical user interface is based upon the Javascript framework `React-JS` using an existing program library `react-crossword`[13]. Fig. 4 depicts the GUI. The library expects a JSON-based puzzle representation that the Prolog back-end creates after a successful puzzle generation. We extended the exemplary use of the library with two more UI elements: "Gib Lösungswort" (give solution for a clue), and "Gehe zu Rover" (go to Rover). The first element looks up the solution in the JSON-based crossword representation, and the second element directs users to a Rover page that shows all the information it has on the word. For this purpose, we augmented the API of Rover to allow such invocations.

A fully functional GUI front-end (currently only used for our simpler crossword generator) is available at `https://vacvvm.eu` (temporary location). As one can see from the screenshot, users have a choice between lexical resources. We have also allowed the crossword generator to make use of Princeton WordNet (Miller, 1995) and the DWDS *Wörterbuch*. For the time being, this version of the software only takes the 132,972 WordNet glosses as input; hypernym or antonym relations are currently not used.[14] Also, only a limited amount of DWDS data is being used. With these two other lexical resources, users get also easy access the PWN GUI, or to the DWDS website to get more information about the word being searched for. In sum, the puzzle GUI hence aims at luring users to other software that can be used to further explore lexical-semantic wordnets, in a sense acquiring more users for those resources.

5.2 Prolog-based puzzle export to PDF

The GUI in Fig. 4 also has an element "Rätsel als PDF", which allows users to download a PDF variant of the puzzle. A Prolog-based converter has been implemented that transforms the internal Prolog representation into LaTeX source code that is automatically compiled into PDF. For this purpose, the LaTeX package `cwpuzzle`[15] has been used, also for the generation of Fig. 3. Usually, the crossword is generated on the front page; its solution is printed on the back page.[16]

[13] `https://github.com/JaredReisinger/react-crossword`

[14] That is, only information from the two files `wn_g.pl` and `wn_s.pl` were used.

[15] `http://www.gerd-neugebauer.de/software`

[16] Once the PDF version has been printed, a comfy armchair is the only other prerequisite to start tackling the crossword.

6 Discussion & Future Work

The automatic generation of crossword puzzles has also been studied in the linguistics community. (Rigutini et al., 2012) present *WebCrow-generation*, a system that does both clue generation and crossword compilation. A large part of their efforts is spent by crawling the Web to extract definitions from text, which can then be used for crosswords. To satisfy the constraints to fill a given puzzle, the authors also borrow the heuristics from (Ginsberg et al., 1990). Also, partially solved puzzles are ranked in terms of their "goodness", *i.e.*, how far a given partial puzzle is from the fully-solved puzzle. The best-ranked puzzle is worked on next.

The use of *existing* lexical information is described by (Aherne and Vogel, 2006). Their system relies on WordNet, and the authors put considerable emphasis on the quality of clue generation with regard to thematic domains such as *Earth* or *Sport*. In the future, we intend to also reduce our lexicon to only contain entries of given thematic domains. In part, this will allow us to investigate how our solver reacts to smaller branching factors, without relying on artificially introduced k values. The use of LLM for clue generation, however, opens up new possibilities as one is not limited to using static information from existing lexical resources.

It is our foremost intention, however, to focus on bringing together and exploiting existing lexical resources for crossword generation. Besides wordnets, thesauri, and dictionaries, we would like to also pursue the idea brought forward by (Smith and Steen, 1981), namely, the use of concordances to generate clues which refer to well-known quotations from plays or books, and where the appropriate word omitted needs to be identified.

Future work is targeted at better understanding an improving our crossword algorithm. Here, we would like to investigate additional heuristics such as giving preference to longer word slots. Ginsberg's hardest test puzzle, which is also the hardest puzzle for our solver, requires four words of length 13. In a first phase, we would like to have our solver to first identify four candidates words (which intersect with each other); and in a second phase use the approach discussed in the paper to solve the rest of the puzzle.

A second line of research concerns clue generation. Anectodical evidence, see Fig. 3, shows that ChatGPT is performing very well in this task. But clearly, a more systematic study is required here, *e.g.*, are automatically generated clues as much fun as humanly generates ones? Can people tell the difference between these two types of clues? Also, how well can we get LLMs to tailor clue generation to specific target audiences?

In a related strand of future work, which is being panned out now, we would like to use the crossword puzzle generator to target both native speakers and second language learners. We aim at investigating how users of both groups play the crossword puzzles: which clues, and the words they hint at, are difficult (within the context of already solved clues)? Is there, for instance, a correlation with word frequencies, or thematic domains? In this respect, our users become part of a citizen science community helping us to better understand language (learning) difficulty.

It shows that large language models (LLM) such as ChatGPT can be used to generate crossword puzzle clues. But given a crossword puzzle such as the one given in Fig. 3, how well do LLMs perform when they are asked to generate solutions words for a given clue? The gold standard for this task is set by the work of Ginsberg and his colleagues on automated crossword solving. Their Berkeley Crossword Solver won first place at the most prestigious human crossword tournament using a combination of neural question answering models, belief propagation and local search (Wallace et al., 2022).

For this other direction, from clues to words, we would like to make use of auto-generated crossword puzzles to fine-tune large language models. We found anecdotical evidence that LMM are surprisingly good at providing help with crossword puzzle clues (that is, generating words described by the clues). But we believe that there is a good opportunity to fine-tune LLM in this respect, in particular, if we want second language learners to not just ask for a crossword cell or slot to be filled, but to engage them in a dialogue that provides scaffolding help. Surely, some clues are better than others to hint at a specific word, but what makes a clue particularly effective in this respect, especially, in the context of second language learners?

The initial motivation of our work was driven by our desire to make a scientific resource such as GermaNet easily available to the lay person. Driven by the popularity of crossword puzzles, we wanted to popularise (and "market") our resource to the general public. The crossword generator will soon appear on our project's website as part of dissemination activities. Crosswords give users a good first

insight into the GermaNet resource; with the Rover web application being invokable from the puzzle for each solution word, users can then explore the wordnet in all dimensions. We invite readers to try-out the crossword generator, recommend it to others, and look forward to their feedback.

7 Ethical Considerations

We do not see any conflict of our work with the principles set out in the ACL Ethics Policy.[17] Our crossword generator makes use of GermaNet and other lexical resources. GermaNet has been constructed over the last 25 years and manually maintained ever since. We are not aware of any discriminatory content. The prototype version of the crossword generator automatically includes ChatGPT-generated clues for words into the puzzle. Such contributions will need to be evaluated in ethical terms before the system goes public.

8 Limitations

The Prolog solver is limited by the lexical resources and computing power at its disposal. As the evaluations show, solving highly interlocked puzzles is by no means trivial and computationally expensive. More work is required to solve more complex grids in less time. Clue generation uses foremost the information from GermaNet. An experimental interface to ChatGPT has been implemented. The quality of the clues, however, need to be carefully evaluated and compared to clues found in humanly-constructed crossword puzzles.

Our evaluation is limited by our small test set of puzzles. To better understand the nature of heuristics, the k value used, and the backtracking mechanism – an excellent discussion is given by (Ginsberg et al., 1990) – we would like to randomly generate puzzles of various interlocking ratios. We believe that the number of clues to solve a given puzzle is less indicative to a problem's hardness than the number of constraints (*i.e.*, the number of word intersections) that need to be observed. In our test set, we see anecdotical evidence for this: a fully interlocked $6x6$ puzzle with 12 word slots is unsolvable (within a given time threshold), but the 13x13 puzzles from Ginsberg's testset with 60 to 64 word slots is solvable. Here, future work is required to better understand the interlocking ratio our heuristic solver can realistically handle.

[17] https://www.aclweb.org/portal/content/acl-code-ethics

9 Acknowledgements

The work has been carried out as part of Text+, the NFDI infrastructure for the Humanities, which is funded by the German Research Foundation (ref. 460033370). – We wish to thank Bettina Braun for her input on the statistical analysis and the reviewers for their helpful comments.

References

A. Aherne and C. Vogel. 2006. Wordnet enhanced automatic crossword generation. In *Proceedings of the Third International Wordnet Conference (Seogwipo, Korea)*, pages 139–145.

H. Berghel. 1987. Crossword compilation with horn clauses. *The Computer Journal*, 30(2):183–188.

H. Berghel and C. Yi. 1989. Crossword Compiler-Compilation. *The Computer Journal*, 32(3):276–280.

M. Ginsberg. 2011. Dr.Fill: Crosswords and an Implemented Solver for Singly Weighted CSPs. *J. Artif. Intell. Res. (JAIR)*, 42:851–886.

M. Ginsberg, M. Frank, M. Halpin, and M. Torrance. 1990. Search Lessons Learned from Crossword Puzzles. *AAAI-90 Proceedings of the Eighth National Conference on Artificial Intelligence*.

B. Hamp and H. Feldweg. 1997. GermaNet - a Lexical-Semantic Net for German. In *Proceedings of the ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*. Madrid, Spain.

V. Henrich, E. Hinrichs, and T. Vodolazova. 2014. Aligning GermaNet Senses with Wiktionary Sense Definitions. In *Human Language Technology: Challenges for Computer Science and Linguistics*, pages 329–342.

G. A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41.

L. Rigutini, M. Diligenti, M. Maggini, and M. Gori. 2012. Automatic Generation of Crossword Puzzles. *Int. J. Artif. Intell. Tools*, 21.

P. D. Smith and S. Y. Steen. 1981. A prototype crossword compiler. *The Computer Journal*, 24(2):107–111.

E. Wallace, N. Tomlin, A. Xu, K. Yang, E. Pathak, M. Ginsberg, and D. Klein. 2022. Automated crossword solving. In *Proceedings of the 60th Annual Meeting of the ACL*, pages 3073–3085, Dublin, Ireland. Association for Computational Linguistics.

C. Zinn, M. Hinrichs, and E. Hinrichs. 2022. Adapting GermaNet for the Semantic Web. In *Proc. of the 18th Conf. on Natural Language Processing (KONVENS 2022)*, pages 41–47, Potsdam, Germany.