FLEXIBLE LEXICAL PROCESSING

Benchmarking DIG for Flexible Lexical Processing

Runtime, energy, coverage, and multi-representation compared with German NLP pipelines and morphological analyzers.

A controlled benchmark of the **Deterministic Inference Graph**: an explicit lexical architecture built around interconnected Hypernodes and relevance-activated inference paths.

MODE Deterministic
OUTPUT Traceable

CENTRAL RESEARCH PROPOSITION

Flexibility without neural prediction.

The benchmark tests whether productive lexical coverage can coexist with deterministic execution, explicit multi-representation, traceability, and computational characteristics.

01 Productive coverage
02 Multi-representation
03 Local path activation
04 Direct traceability

STUDY PROFILE

STUDY ID
260930-LEX1.1

TASK
German morphological analysis

PROTOCOL
50 isolated word forms · 10,000 repetitions

MEASUREMENT DATE
30 September 2026

CeLLAI-Team
CeLLAI FLEXCO · Vienna, Austria

THE BENCHMARK IN ONE VIEW

DIG Small achieves zero observed missing words with major runtime and energy advantages.

17–52×
FASTER THAN THE EVALUATED NEURAL NLP PIPELINES

38–164×
LESS NET ENERGY PER WORD

5.34 µs
RUNTIME / WORD
187,149 words per second; 17x faster than the fastest NLP comparator.

0.23 mJ
NET ENERGY / WORD
The evaluated neural pipelines require 38–164× the net energy per word of DIG Small.

0 / 50
MISSING WORDS
DIG Small and DIG Big returned analyses for every tested surface form.

Explicit
MULTI-REPRESENTATION
Surface, lemma, morphology, inflection, derivation, compounds, and etymology.

Runtime and net energy across all evaluated systems

bars are logarithmically scaled; exact values are shown

MODEL	RUNTIME / WORD (MICROSECONDS)	NET ENERGY / WORD (mJ)
DIG Small	5.34	0.23
DIG Big	8.89	0.27
GiellaLT	11.74	0.44
DWDSmor	16.28	0.71
DEMorphy	83.94	1.27
spaCy Small	92.98	8.71
spaCy Large	97.81	9.26
spaCy Transformer	238.38	37.70
Stanza	279.25	29.58

DIG Inference graph · FST Finite-state · NLP Neural pipeline

THE ARCHITECTURAL PITCH

A third axis for linguistic computation.

DIG investigates whether the flexibility expected from NLP can be achieved through deterministic compositional construction over an explicit lexical graph.

DETERMINISTIC · PRODUCTIVE · TRACEABLE · EFFICIENT

Initial evidence from the present lexical benchmark.

SCOPE & CAUTION

Strong efficiency signal. Preliminary linguistic evidence.

- 50 manually curated isolated forms
- 10,000 repetitions
- Same workstation; best available execution mode
- No complete gold-standard accuracy evaluation yet

Next: held-out lexical productivity, formal error classes, larger datasets, sentence context, and scaling behavior.

DOCUMENT MAP

Contents

Experimental design, system architecture, benchmark protocol, comparative results, limitations, and the validation roadmap for deterministic lexical inference.

STUDY **260930-LEX1.1** MEASURED **07 AUG 2026**

01 Introduction and Summary 1

02 General Experiment Description 2

03 Hardware and Software Specifications 3

03.1 Hardware Specifications 3

03.2 Software Specifications 3

04 Model Specifications 3

04.1 Model Descriptions 3

04.2 Further Model Information for Reproducibility 6

05 Dataset and Gold-Standard Specifications 6

05.1 Dataset 6

06 Measurements 6

06.1 Measurement Protocol 6

06.2 Measurement Tools 7

07 Overall Comparison Results 9

08 Conclusions 10

08.1 The Central Result: Flexibility Without Neural Prediction 10

08.2 Why the NLP Pipelines Are the More Relevant Conceptual Comparison 11

08.3 Coverage, Missing Words and Incorrect Readings 11

08.4 Computational Efficiency 12

08.5 DIG Small and DIG Big 12

08.6 A Possible Third Axis of Linguistic Processing 13

08.7 Limitations of the Present Evidence 13

08.8 Overall Conclusion 14

09 Future prospects 15

09.1 RAM Optimization as the Principal Engineering Priority 15

09.2 Separating the DIG Architecture from the Current Implementation 15

09.3 Next-Generation Runtime and Energy Targets 16

09.4 A Dedicated Productivity and Coverage Benchmark 16

09.5 Scaling With Word Length and Morphological Complexity 17

09.6 Replacing the Preliminary Unsupported-Analysis Metric 18

09.7 Normalizing Comparisons by Output Richness 18

09.8 From Isolated Words to Sentences 19

A Appendix: 50-Word Test Set 20

Scientific References 22

01 Introduction and Summary

> Modern linguistic processing is commonly divided between two fundamentally different approaches [1–4]. Deterministic finite-state analyzers offer efficiency and reproducibility through predefined lexical resources, whereas neural NLP systems extend flexible processing to variable or previously unseen input at substantially greater computational cost and with less direct traceability at the level of individual lexical inference.

DIG (Deterministic Inference Graph) introduces a different processing principle. It is neither a conventional morphological automaton nor a neural language model. Instead, DIG is designed as a deterministic, compositional and directly interpretable inference architecture. The objective is not to enumerate every possible surface word in advance, but to construct linguistic representations productively from a comparatively compact lexical foundation.

This distinction is central to understanding the benchmark. DIG is designed as a **flexible word-building system**. The intended lexical foundation of DIG is based on approximately 151,000 lexical core entries, corresponding in scale to the roughly 151,000 headwords (Grundformen) contained in the 29th edition of *Duden – Die deutsche Rechtschreibung* [5, 6]. From this comparatively compact lexical basis, DIG is designed to construct additional inflected, derived, and compound word representations through deterministic compositional inference. Consequently, increasing linguistic coverage does not conceptually require storing every possible German word form as an independent lexical entry. Instead, lexical knowledge can be recombined through deterministic inference paths.

This changes the relevant comparison. Finite-state analyzers such as DWDSmor, DEMorphy and GiellaLT remain important benchmarks [7–10] because they provide strong deterministic references for computational efficiency, reproducibility and interpretability. The NLP pipelines constitute the more informative conceptual comparison with respect to coverage and flexible processing, whereas the finite-state systems provide the more direct comparison with respect to determinism and computational efficiency. The present benchmark illustrates this distinction directly. **DIG Small and DIG Big produced no missing words across the evaluated dataset. The evaluated spaCy and Stanza pipelines likewise produced no missing words, whereas the finite-state analyzers produced missing entries.** The important question is therefore not simply whether DIG can outperform an automaton. It is whether the flexibility usually associated with modern NLP can be achieved through an entirely different computational mechanism: deterministic graph inference rather than neural prediction.

Against this architectural background, the efficiency results become particularly significant. DIG Small processes the benchmark at **5.34 µs per word**, compared with 92.98 µs for spaCy Small, 97.81 µs for spaCy Large, 238.38 µs for spaCy Transformer and 279.25 µs for Stanza. Under the present benchmark conditions, this corresponds to an approximately **17- to 52-fold runtime difference** between DIG Small and the evaluated neural NLP pipelines. Net energy consumption shows an even larger separation: DIG Small requires **0.23 mJ per word**, compared with 8.71 mJ for spaCy Small, 9.26 mJ for spaCy Large, 37.70 mJ for spaCy Transformer and 29.58 mJ for Stanza.

The central result of this benchmark is therefore not simply that **DIG is fast**. The more consequential hypothesis is that DIG may establish a **third axis for linguistic data processing**: combining the productive flexibility and lexical coverage expected from NLP systems with deterministic execution, explicit multi-representation, traceability and computational characteristics closer to highly optimized symbolic systems.

In this sense, DIG should not be understood as another morphological automaton. Its architectural proposition is more fundamental: **linguistic flexibility does not necessarily require neural prediction, and deterministic processing does not necessarily require sacrificing productive lexical coverage.** The

present benchmark provides initial empirical evidence for this proposition. A larger gold-standard evaluation will be required to establish its general linguistic accuracy and scalability, but the current results already indicate that the efficiency–flexibility trade-off traditionally assumed between symbolic analyzers and neural NLP systems may not be the only available design space.

KEY STATEMENT

The benchmark tests a specific architectural hypothesis: *productive lexical flexibility and deterministic, traceable inference need not be mutually exclusive.*

Source-integrity note. The supplied bibliography explicitly marks the exact Duden dataset / export underlying the approximately 151,000 core Hypernodes as still to be specified. Accordingly, the current text cites the 29th Duden edition and German word-formation literature for scale and linguistic background, but it does not invent a dataset citation.

02 General Experiment Description

Table 1 General experiment description

FIELD	ENTRY
Study / Experiment ID	260807-LEX1.1
Task	Morphological analysis of German word forms
Comparison	German morphological analyzers and NLP pipelines versus DIG
Language	German, with Old Greek, Latin, French, and English lexical material represented in the corpus
Input data	50 isolated German word forms
Primary metrics	Runtime and energy consumption
Additional metrics	GPU energy, CPU energy, peak RAM, and peak reserved VRAM
Execution mode	Local execution on the same workstation using each system's best available configuration
Repetitions per model	10,000
Measurement date	07 August 2026

All benchmark values reported in this manuscript are internal experimental results. External references document the evaluated systems, measurement tools, and scientific background rather than the paper's own measurements.

03 Hardware and Software Specifications

03.1 Hardware Specifications

Table 2 Hardware configuration

COMPONENT	SPECIFICATION
Workstation	Lenovo ThinkStation P8; 1,000 W system power limit
CPU	AMD Ryzen Threadripper PRO 9975WX; 32 cores / 64 threads; 4 GHz base clock; reported boost 5.484 GHz
GPU	NVIDIA RTX PRO 5000 72GB Blackwell; 73,415 MiB VRAM; 300 W power limit; single GPU
System memory	128 GB SK hynix DDR5 (4 × 32 GB), 6,400 MT/s

03.2 Software Specifications

Table 3 Software configuration

SOFTWARE	VERSION / CONFIGURATION
Operating system	Debian GNU/Linux 13.5 Trixie; kernel 6.12.95+deb13-amd64
GPU driver / CUDA	NVIDIA driver 595.84; driver-supported CUDA 13.2
Python	3.13.5 and 3.9[a]
PyTorch	2.11.0; PyTorch CUDA 13.0
CuPy	14.1.1

[a] Python 3.9 is used only for DEMorphy and GiellaLT because those systems are not compatible with Python 3.13 in the reported setup.

04 Model Specifications

04.1 Model Descriptions

> The comparison is deliberately restricted to German-language systems and resources so that architectural differences are not confounded by language-specific model variation. The underlying lexical material also contains foreign lexical elements, including Old Greek, Latin, French, and English forms, reflecting the composition of the corpus used by DIG.

No existing system is directly equivalent to the proposed architecture. The benchmark therefore spans two established comparison classes: deterministic finite-state morphology and modern neural NLP pipelines. The present report is primarily an efficiency and behavior benchmark; a complete gold-standard accuracy comparison remains future work. Missing analyses and preliminary unsupported-analysis counts are reported as descriptive indicators rather than as substitutes for a validated accuracy metric.

Proposed architecture

DIG is a Deterministic Inference Graph that transforms raw text into an explicit lexical representation. Two configurations are evaluated: DIG Small with approximately 600,000 Hypernodes and DIG Big with approximately 2.4 million Hypernodes. The present dataset does not demonstrate a coverage advantage for the larger graph; its inclusion instead provides an initial indication of how graph size affects runtime, energy, and memory.

Deterministic finite-state baselines

Deterministic finite-state analyzers are included because they represent a fundamentally different approach to morphological analysis from NLP models [1, 2, 10]. Their analyses are obtained through deterministic lookup and finite-state transduction, resulting in highly interpretable and reproducible outputs. They additionally serve as strong references for runtime efficiency and output consistency because they provide deterministic outputs and serve as useful efficiency and reproducibility baselines.

- DWDSmor is a finite-state morphological analyzer and generator for written German. It is implemented using Stuttgart Finite-State Transducers (SFST), which map surface word forms to their corresponding lemmas and morphosyntactic analyses. The analyzer produces deterministic lexical analyses, including lemma, part-of-speech information, and morphological features where available [1, 2, 7, 11].
- DEMorphy is a German morphological analyzer based on a Deterministic Acyclic Finite-State Automaton (DAFSA). It performs efficient lexical lookup and returns lemma and morphological analyses for input words. Its dictionary-based approach provides deterministic and interpretable analyses with low computational overhead [8].
- GiellaLT is a finite-state linguistic framework based on Helsinki Finite-State Technology (HFST) and constraint grammar. Although German support is currently in an early development stage, it is included as an example of another deterministic finite-state approach to morphological analysis. GiellaLT supports multiple languages. In this report we report the results for the German model [9, 10, 12].

Interpretive note. In the present dataset, the finite-state baselines preserve deterministic behavior but exhibit missing analyses.

Neural NLP pipelines

- spaCy is a widely used NLP library providing efficient tokenization, part-of-speech tagging, lemmatization, morphological analysis, and dependency parsing [3]. For this evaluation, three German pipelines were considered: de_core_news_sm, de_core_news_lg, and de_dep_news_trf. In this report, these models are referred to as spaCy small, spaCy large and spaCy Transformer. The small pipeline was included as a lightweight baseline for runtime and energy efficiency comparisons, while the large and transformer-based pipelines represent higher-capacity models designed to maximize linguistic accuracy. Furthermore, we excluded the NER and parser components in each spaCy pipeline because these tasks are outside the scope of the comparison.
- **Stanza:** A neural NLP toolkit that provides linguistic preprocessing, including tokenization, part-of-speech tagging, lemmatization, morphological feature prediction, and syntactic parsing. Stanza supports many languages, but each language is processed through a separate pipeline. Therefore, the German model is independent of models for other languages. Its models were trained on Universal Dependencies treebanks and were designed to maximize linguistic accuracy across multiple languages, making it an appropriate benchmark for evaluating the quality of lexical and grammatical analysis. For this evaluation, the Stanza pipeline included the tokenize, pos, and lemma processors [4]

Interpretive note. The neural pipelines provide broad output coverage, but the current benchmark also records unsupported analyses under its preliminary operational definition.

Table 4 Architectural and output characteristics of the evaluated systems

MODEL	ARCHITECTURE	TOKEN	MORPH.	ALT.	ADDITIONAL OUTPUT / CAPABILITY
DIG	Deterministic Inference Graph	✓	✓	✓	Inflections; adjective syntactic function; etymology; compound and derivational information
DWDSmor	Finite-state automaton (SFST)	—	✓	✓	Inflection class; adjective syntactic function
DEMorphy	Finite-state automaton (DAFSA)	—	✓	✓	Adjective syntactic function
GiellaLT	Finite-state automaton (HFST)	—	✓	✓	Adjective syntactic function
spaCy Small	CNN-based	✓	✓	—	NER and dependency parsing available in the broader pipeline
spaCy Large	CNN + word embeddings	✓	✓	—	NER and dependency parsing available in the broader pipeline
spaCy Transformer	Transformer	✓	✓	—	NER and dependency parsing available in the broader pipeline
Stanza	Neural network components	✓	✓	—	NER and dependency parsing available in the broader pipeline

Token representation follows the manuscript's comparison definition. DIGs tokenization strategy differs from the conventional notion used in neural architectures.

For models that support batched inference, the entire dataset is processed as a single batch to improve throughput. Likewise, when GPU acceleration is available, the GPU-enabled backend is used to reduce inference time. The GPU-based pipelines return fully materialized CPU-side output objects rather than unfinished GPU-resident tensors. The models supporting both batched inference and GPU acceleration are spaCy models (de_core_news_sm, de_core_news_lg, and de_dep_news_trf) and Stanza. The remaining models support neither batched inference nor GPU acceleration and are therefore evaluated using sequential CPU-based inference. Consequently, the benchmark evaluates each system using its best available execution configuration rather than under identical execution conditions.

04.2 Further Model Information for Reproducibility

The version numbers of models are given below:

Table 5 Model versions used in the benchmark

MODEL	VERSION
DIG	1.1.0
DWDSmor	0.18.0
DEMorphy	1.0
GiellaLT	0.2.0
spaCy Small	3.8.0
spaCy Large	3.8.0
spaCy Transformer	3.8.0
Stanza	1.14.0

05 Dataset and Gold-Standard Specifications

05.1 Dataset

> The evaluation set contains 50 manually curated German word forms, each processed as an isolated surface form without sentential context. The items cover distinct inflectional and grammatical phenomena, and each system is expected to return the grammatically valid analyses available for a given form. The complete list is reproduced in Appendix A; the grammatical reference framework is supported by standard German grammar and word-formation sources [5, 6, 13].

Gold-standard status. This set of wordlist was randomly produced by ChatGBT 5.5. The supplied reference file states that the gold standard was validated. The appendix therefore reproduces the manuscript's test material as provided.

06 Measurements

06.1 Measurement Protocol

> All systems were evaluated on the same hardware environment while using their best available execution configuration. Each model was measured over 10,000 repetitions to reduce the influence of runtime noise and short-term system variation.

Before the timed measurements, each model performed 100 warm-up runs. Warm-up runs, model loading, and initialization were excluded from the measurement process, thus only the inference phase was evaluated.

Runtime measurements started immediately before the first input word was passed to the loaded model and ended immediately after the model produced its outputs for the complete dataset. Between repetitions, GPU memory was cleared to minimize carry-over effects.

All experiments were conducted on a dedicated workstation without competing background workloads. Furthermore, the testing order was randomized to reduce potential systematic bias.

The experimental setup is summarized in the table below:

Table 6 Measurement protocol

PARAMETER	SETTING
Repetitions per model	10,000
Warm-up runs	100
Warm-up included in measurement	No
Model loading included	No
Measurement start	Immediately before the first input is passed to the loaded model
Measurement end	Immediately after model inference completes
Execution environment	Dedicated workstation with no competing workloads
Run isolation	Each benchmark block executed in an independent process
Execution order	Randomized

06.2 Measurement Tools

All benchmark measurements were conducted using a dedicated shell script. To minimize measurement overhead and avoid interference between pipelines, the script invoked one pipeline and one timed run at a time. For each metric, the corresponding measurement was obtained immediately before and after the respective run. The models were executed in a randomized sequential order to reduce potential bias caused by execution order.

To estimate the energy consumption attributable to each model, we first determine the baseline power consumption of the evaluation system. Energy-aware reporting and inference benchmarking are established concerns in NLP and machine learning [14–17]. The idle power draw of the evaluation machine was measured as 131 W on average.

However, the benchmark script itself introduces an additional power overhead, even when no model is being executed. To account for this baseline load, we measured the power consumption during idle runs of the benchmark script, in which the script remained active but paused instead of executing a model. Under these conditions the system consumed an average power draw of **137.69 W** on average. This value was used as the baseline power consumption and subtracted from the measured system power when calculating the net energy consumption attributable to each model.

The net energy consumption of a benchmark run is then computed by subtracting the estimated idle energy from the total measured energy:

$$E_{\text{net}} = E_{\text{total}} - P_{\text{idle}} \times \Delta t$$

where E_{total} is the total energy consumed during the benchmark execution, P_{idle} is the average idle power of the system running the script in idle, and Δt is the execution time of the benchmark. The resulting value represents the estimated energy consumed by the evaluated model alone.

Table 7 Measurement tools and methods

METRIC	DEVICE / TOOL	METHOD	UNIT
Runtime	Python `time.perf_counter()` [18]	End timestamp minus start timestamp	ms
System energy	Shelly Pro 2PM external smart power meter [19]	End cumulative energy minus start cumulative energy	J
GPU energy	NVIDIA Management Library (PyNVML) [20]	End cumulative energy minus start cumulative energy	J
CPU energy	Linux Powercap interface [21]	End cumulative energy minus start cumulative energy	J
Peak process RAM	OS process statistics (`resource.getrusage()`)	Maximum resident memory used by the benchmark process	MiB
Peak reserved VRAM	PyTorch CUDA or CuPy memory API [22, 23]	Maximum GPU memory reserved by the benchmark process	MiB

Single inferences are too short for stable external energy measurement. The experiment therefore aggregates 10,000 dataset repetitions per measurement block. Reported peak-reserved memory may include benchmark-script overhead; the GPU memory backend depends on the framework used by each evaluated model.

07 Overall Comparison Results

The primary comparison is reported per input word so that systems with substantially different total runtimes remain directly comparable. Runtime, throughput, net external energy, coverage behavior, and preliminary unsupported-analysis counts are separated from subsystem power and memory measurements to avoid visually conflating distinct metrics.

Table 8 Primary performance and coverage results

MODEL	RUNTIME (µs / WORD)	THROUGHPUT (WORDS / S)	NET ENERGY (mJ / WORD)	MISSING	UNSUPPORTED
DIG Small	**5.34**	**187,148.98**	**0.23**	**0**	**2**
DIG Big	**8.89**	**112,519.63**	**0.27**	**0**	**2**
DWDSmor	16.28	61,413.90	0.71	2	0
DEMorphy	83.94	11,913.24	1.27	1	0
GiellaLT	11.74	85,167.61	0.44	11	0
spaCy Small	92.98	10,754.43	8.71	0	18
spaCy Large	97.81	10,224.09	9.26	0	16
spaCy Transformer	238.38	4,195.01	37.70	0	15
Stanza	279.25	3,581.06	29.58	0	15

"Unsupported" is the benchmark's preliminary operational category for documentable incorrect readings. It is not presented as a finalized gold-standard accuracy metric.

KEY STATEMENT

DIG Small records 5.34 µs per word and 0.23 mJ net energy per word while returning analyses for all 50 evaluated forms. The principal comparison is therefore not speed alone, but the combination of coverage behavior, deterministic execution, and resource use.

Table 9 Energy measurements across the complete 10,000-repetition benchmark block

MODEL	GROSS SYSTEM ENERGY (J)	NET SYSTEM ENERGY (J)	GPU ENERGY (J)	CPU ENERGY (J)
DIG Small	502.56	**113.19**	42.00	216.33
DIG Big	725.04	**134.70**	51.17	328.87
DWDSmor	1,474.56	353.56	131.74	627.82
DEMorphy	6,414.48	635.62	535.94	2,794.67
GiellaLT	1,026.00	217.65	90.75	459.47
spaCy Small	10,755.00	4,353.45	3,242.23	3,599.47
spaCy Large	11,362.32	4,628.71	3,495.25	3,791.47
spaCy Transformer	35,262.72	18,851.56	15,406.04	9,367.34
Stanza	34,012.80	14,788.03	11,340.80	10,585.83

GPU and CPU energy are subsystem measurements and must not be algebraically added to the net external system-energy value. Net system energy is derived from the external Shelly measurement after baseline correction.

Table 10 Power draw and memory footprint

MODEL	SYSTEM POWER (W)	CPU POWER (W)	GPU POWER (W)	PEAK RAM (MiB)	PEAK VRAM (MiB)
DIG Small	188.11	80.97	15.72	4,442.10	**0.00**
DIG Big	163.16	74.01	**11.52**	17,642.66	**0.00**
DWDSmor	181.12	77.11	16.18	157.14	0.00
DEMorphy	**152.83**	**66.59**	12.77	392.96	0.00
GiellaLT	174.76	78.26	15.46	**110.00**	0.00
spaCy Small	231.33	77.42	69.74	1,041.11	10.83
spaCy Large	232.34	77.53	71.47	2,568.31	583.18
spaCy Transformer	295.86	78.59	129.26	2,128.88	506.00
Stanza	243.60	75.82	81.22	1,847.53	254.00

Zero VRAM for the deterministic systems reflects the reported benchmark configuration. DIG's present RAM footprint, particularly for DIG Big, remains an engineering optimization target rather than a claimed architectural minimum.

08 Conclusions

> The purpose of the benchmark is not merely to identify the fastest processor of isolated German word forms. It asks whether a different architecture can combine properties that are usually separated across existing approaches: productive lexical flexibility, broad coverage, deterministic processing, explicit multi-representation, interpretability, traceability, and low computational cost. Under the present conditions, DIG provides initial evidence that such a combination is possible.

08.1 The Central Result: Flexibility Without Neural Prediction

The most important result of this study is not the runtime advantage of DIG in isolation. It is that this efficiency was achieved by a system that is designed to remain **lexically productive rather than being restricted to the recognition of a predefined inventory of complete word forms**.

DIG is not a finite-state morphological automaton. Its architecture is based on interconnected Hypernodes and deterministic inference paths through which lexical structures can be composed and interpreted. The long-term design principle is that a comparatively compact lexical foundation—approximately 151,000 core Hypernodes corresponding to the underlying Duden-based corpus—can serve as the basis from which additional inflected, derived and compound word forms are constructed.

This distinction is fundamental. A system does not become flexible merely because it contains a sufficiently large dictionary. The relevant property is whether linguistic structures can be **productively generated and interpreted from reusable components**. DIG is designed around precisely this principle.

The present benchmark provides an initial indication of this capability. Both DIG Small and DIG Big returned analyses for all 50 tested word forms and therefore produced **zero missing words** in the evaluated dataset. The neural NLP pipelines also produced zero missing words, whereas the evaluated finite-state analyzers showed missing entries.

The result therefore points toward an important architectural hypothesis:

Deterministic processing does not necessarily require sacrificing productive lexical coverage.

If this property remains stable on substantially larger and more systematically constructed datasets, it would distinguish DIG from conventional deterministic lexical lookup architectures in a fundamental way.

08.2 Why the NLP Pipelines Are the More Relevant Conceptual Comparison

DWDSmor, DEMorphy and GiellaLT remain important benchmark systems. They provide strong deterministic reference points for runtime, reproducibility and computational efficiency. For that reason, they are valuable technical baselines.

However, they are not necessarily the closest conceptual comparison to DIG.

The central capability DIG is intended to provide is **flexible linguistic processing beyond a fixed inventory of surface forms**. In that respect, spaCy and Stanza address a more similar practical problem. Neural NLP systems are designed to continue producing analyses for variable and previously unseen linguistic input rather than simply returning no result when a complete lexical form is absent from an encoded resource.

The present results illustrate this difference. All evaluated NLP pipelines returned outputs for every word in the dataset, as did DIG. The deterministic finite-state systems, by contrast, produced between one and eleven missing words.

For this reason, the most informative question is not:

Can DIG compete with a morphological automaton?

The stronger question is:

Can DIG provide the flexibility and coverage expected from modern NLP while replacing probabilistic neural inference with deterministic, explicit and traceable lexical construction?

From this perspective, the comparison with spaCy and Stanza becomes particularly significant.

DIG Small requires **5.34 µs per word**, while the evaluated NLP pipelines require 92.98 µs for spaCy Small, 97.81 µs for spaCy Large, 238.38 µs for spaCy Transformer and 279.25 µs for Stanza. The resulting difference ranges from approximately **17×to 52×in runtime**.

The energy measurements show an even larger separation. DIG Small requires **0.23 mJ of net energy per word**, compared with 8.71 mJ for spaCy Small, 9.26 mJ for spaCy Large, 37.70 mJ for spaCy Transformer and 29.58 mJ for Stanza.

The relevance of these differences is therefore not simply that one implementation happens to execute faster than another. The result suggests that **flexible lexical processing may be achievable through a computational mechanism with fundamentally different resource requirements from contemporary neural NLP pipelines**.

08.3 Coverage, Missing Words and Incorrect Readings

The observed error patterns reveal another potentially important distinction between the three architectural classes.

In the present dataset, the deterministic finite-state systems showed **missing words but no documented unsupported analyses**. The neural NLP systems showed **no missing words but substantially more documented incorrect readings under the current benchmark definition**. DIG occupied a different position: it showed **zero missing words and two documented incorrect readings** for each configuration.

This pattern is potentially more informative than runtime alone.

The conventional deterministic solution appears to favor strict reproducibility at the risk of lexical non-coverage. Neural systems favor coverage and flexibility but may generate analyses that are not supported by

the intended lexical interpretation. DIG is intended to combine the two characteristics: productive coverage together with deterministic inference.

However, these observations must remain preliminary. The current benchmark does not yet contain a complete gold-standard linguistic accuracy evaluation, and the operational definition of an unsupported analysis must be formalized more rigorously before strong accuracy claims can be made.

Future evaluations should therefore distinguish at least between:

- completely missing analyses,
- incorrect lemmas,
- incorrect morphological features,
- invalid lexical interpretations,
- partially correct representations,
- valid but unexpected alternative readings,
- and genuinely unsupported generated structures.

Such a distinction will be essential for determining whether DIG's observed behavior constitutes a systematically different error profile from neural NLP systems.

08.4 Computational Efficiency

Even when interpreted conservatively, the computational results are substantial.

DIG Small is the fastest system in the benchmark at **5.34 µs per word**, corresponding to approximately **187,149 processed words per second**. DIG Big reaches 8.89 µs per word.

The fastest non-DIG system, GiellaLT, requires 11.74 µs per word. DIG Small therefore processes the present benchmark approximately **2.2 times faster than the fastest non-DIG comparator**.

Against the neural NLP systems, the difference is considerably larger.

Energy consumption follows a similar pattern. DIG Small records **0.23 mJ net energy per word**, while DIG Big requires 0.27 mJ. The best non-DIG value is 0.44 mJ for GiellaLT. The evaluated neural pipelines range from 8.71 to 37.70 mJ per word.

These results indicate that the computational advantage of DIG cannot be explained exclusively by avoiding transformer-scale neural networks. DIG also performs favorably against lightweight deterministic systems.

This is important because the architectural proposition would be substantially weaker if flexibility were achieved only by accepting the computational cost of modern NLP. The present data instead suggest that the flexible lexical architecture and computational efficiency of DIG may coexist.

08.5 DIG Small and DIG Big

The comparison between DIG Small and DIG Big also provides useful information about the architecture itself.

DIG Small contains approximately 600,000 Hypernodes, while DIG Big contains approximately 2.4 million. Despite this fourfold difference in graph size, the benchmark does not show a corresponding improvement in the measured coverage of the present dataset: both configurations return zero missing words and the same number of currently documented incorrect readings.

At the same time, DIG Big requires greater runtime, energy and particularly RAM.

This makes DIG Small the more efficient configuration for the present test. However, this should not be interpreted as evidence that the larger graph has no purpose. The 50-word dataset is too small to establish whether the additional Hypernodes provide advantages for broader vocabulary coverage, richer lexical relations, additional languages, uncommon derivations, compound structures or deeper inference paths.

A future benchmark should therefore evaluate DIG configurations as a scaling experiment rather than merely as two implementations. The relevant question is how lexical capability, representational richness, runtime and memory consumption change as the Hypernode graph grows.

Such an experiment may also determine how far the graph can be reduced while preserving productive lexical behavior. This is particularly relevant to the proposed architecture in which a substantially smaller core graph could serve as the productive basis for a much larger effective lexical space.

08.6 A Possible Third Axis of Linguistic Processing

Taken together, the findings suggest that DIG should not be positioned simply between finite-state analyzers and neural NLP models on a one-dimensional scale.

The architecture potentially defines a **different axis of linguistic computation**.

Finite-state systems demonstrate that linguistic processing can be deterministic, reproducible and computationally inexpensive.

Neural NLP systems demonstrate that linguistic processing can remain flexible and provide outputs for input that extends beyond explicitly enumerated lexical entries.

DIG investigates whether these properties can be combined through another mechanism:

deterministic compositional inference over an explicit, interconnected lexical representation.

If successful at larger scale, this would represent more than an incremental improvement in morphological analysis. It would demonstrate that the established choice between rigid symbolic lookup and flexible statistical inference is not exhaustive.

A deterministic system could remain productive.

A flexible system could remain traceable.

A broad lexical representation could be constructed without globally activating a neural network.

And linguistic knowledge could potentially be expanded through structured composition rather than through either exhaustive surface-form enumeration or probabilistic parameter learning.

This is the broader significance of the present benchmark.

08.7 Limitations of the Present Evidence

These conclusions must nevertheless be interpreted within the limits of the experiment.

The dataset contains only **50 manually curated isolated word forms**. This is sufficient for an initial controlled efficiency benchmark but not for demonstrating general lexical coverage or linguistic accuracy.

The words were processed without sentential context. Consequently, the experiment does not evaluate contextual disambiguation, sentence-level syntax, dependency structure or discourse-level interpretation.

Furthermore, a comprehensive gold-standard accuracy evaluation has not yet been conducted. The missing-word and incorrect-reading counts provide useful preliminary observations but cannot replace systematic evaluation against independently annotated linguistic reference data.

The compared systems also do not perform identical computations and do not return identical representations. The NLP pipelines and deterministic analyzers therefore answer partly different linguistic questions, while DIG itself produces a broader lexical representation than some of the benchmark systems.

Finally, the systems were executed using their best available configurations rather than an artificially identical computational mode. Neural pipelines could use batching and GPU acceleration where supported, while other systems were executed sequentially on CPU. The measurements therefore represent practical inference performance on the same workstation rather than an attempt to force fundamentally different architectures into identical execution strategies.

These limitations restrict the generality of the present claims, but they do not eliminate the architectural observation established by this first experiment.

08.8 Overall Conclusion

The principal result of this benchmark can therefore be summarized as follows:

DIG demonstrates that, within the present test, deterministic lexical processing can combine zero observed missing words, explicit multi-representation and productive lexical behavior with computational efficiency substantially exceeding the evaluated neural NLP pipelines.

The finite-state analyzers remain important references because they demonstrate the efficiency and reproducibility achievable through deterministic linguistic processing. However, the more consequential comparison for DIG is with flexible NLP systems, because DIG is intended to address the same fundamental requirement: the ability to continue constructing meaningful linguistic representations beyond a fixed list of complete stored word forms.

The decisive difference is how this flexibility is achieved.

Neural NLP systems obtain flexibility through learned statistical representations and probabilistic inference.

DIG attempts to obtain it through **deterministic compositional construction over an explicit lexical graph**.

The current benchmark cannot yet establish that this principle generalizes to unrestricted German vocabulary or to complete natural-language processing. That will require substantially larger datasets, stratified tests of compounds and derivations, systematic evaluation of previously unseen forms, sentence-level experiments and a rigorous gold-standard accuracy study.

Nevertheless, the present results justify a stronger research hypothesis than simply stating that DIG is faster:

KEY STATEMENT

> **DIG may constitute a computational design space in which lexical flexibility, deterministic inference, explicit multi-representation, and high computational efficiency are treated as properties of the same architecture rather than as competing objectives.**

Establishing whether this observation remains valid at lexical, sentence and corpus scale is therefore the central objective for the next stage of evaluation.

09 Future prospects

> The next stage of work must determine whether the architectural pattern observed here survives scale. The objective is no longer simply to demonstrate computational efficiency, but to establish whether productive lexical processing can retain coverage, representational richness, deterministic behavior, and efficiency as the graph is reduced, expanded, and applied to substantially more complex linguistic input.

09.1 RAM Optimization as the Principal Engineering Priority

The most immediate engineering limitation identified by the present benchmark is memory consumption.

This is particularly visible for DIG Big, but memory optimization is also relevant to DIG Small. The current representation has not yet been optimized specifically for minimal memory footprint, and therefore the measured RAM requirements should not automatically be interpreted as an inherent requirement of the DIG architecture.

A principal objective of the next implementation stage is to reduce operational memory consumption toward approximately **1.5–2 GB while retaining the flexibility of the system**.

This optimization is expected to involve several levels of the implementation, including:

- reduced object overhead,
- improved memory locality,
- more efficient indexing structures,
- elimination of redundant lexical information,
- and a stronger transfer of performance-critical operations from Python into optimized native code.

The central requirement is that memory reduction must **not be achieved by sacrificing productive lexical capability**. A smaller DIG that behaves merely as a restricted dictionary would not represent an architectural improvement.

The objective is therefore simultaneous optimization:

less stored structure, less memory, faster traversal, lower energy consumption, while preserving productive lexical construction and multi-representation.

09.2 Separating the DIG Architecture from the Current Implementation

The present implementation combines Python with C++ libraries. This is practical for research and rapid development, but it also makes it difficult to determine which part of the measured computational cost is intrinsic to the DIG architecture and which part results from implementation overhead.

Future development should therefore increasingly separate these two questions.

The scientific question is:

What is the computational complexity of deterministic graph inference itself?

The engineering question is:

How efficiently can this architecture be implemented on current hardware?

These are related but not identical.

Python object management, memory allocation, interpreter overhead, data conversion and non-optimal memory layouts may currently contribute to runtime and RAM consumption independently of the underlying inference principle.

A more mature implementation should therefore move increasingly performance-critical graph operations into optimized native components while maintaining the higher-level flexibility required for research and development.

This would provide a clearer measurement of the architecture itself and make it possible to determine whether further efficiency gains result from better software engineering or from changes to the DIG model.

09.3 Next-Generation Runtime and Energy Targets

The current development roadmap targets a reduction of operational RAM consumption toward approximately 1.5–2 GB.

Associated preliminary engineering estimates suggest that a substantially optimized DIG implementation may be capable of reducing the benchmark runtime toward approximately **1,000–1,500 ms total runtime** and net energy consumption toward approximately **85–95 J** under a comparable experimental configuration.

These values must currently be treated as **engineering targets rather than benchmark results**.

The next benchmark should therefore explicitly test whether these improvements can actually be achieved and, more importantly, which optimization produces each improvement.

This distinction is essential. A reduction in memory consumption may improve cache locality and reduce traversal overhead, but the relationship between memory, runtime and energy consumption must be demonstrated experimentally rather than assumed.

Future optimization experiments should therefore measure each substantial architectural or implementation change independently.

The objective is not merely to report a faster final number. It is to determine **why DIG becomes faster and more energy efficient**.

09.4 A Dedicated Productivity and Coverage Benchmark

The 50-word benchmark was appropriate for an initial controlled comparison, but it cannot demonstrate the principal lexical claim of DIG.

The next experiment should therefore contain a dedicated **productivity and coverage benchmark** designed specifically to test whether DIG can construct valid representations for forms that are not individually stored as complete lexical entries.

Such a benchmark should distinguish several linguistic categories, including:

- regular inflected forms,
- irregular inflected forms,
- derivational morphology,
- short compounds,
- long compounds,
- recursively constructed compounds,

- uncommon but linguistically valid formations,
- foreign lexical elements integrated into German structures,
- ambiguous surface forms,
- and deliberately constructed previously unseen words.

Particularly important would be a controlled **held-out lexical experiment**.

Certain valid forms or compounds should intentionally be absent as complete stored structures from the evaluated graph. The benchmark would then test whether DIG can reconstruct their correct lexical representation from the available Hypernodes and relationships.

Such an experiment would provide much stronger evidence for productive lexical processing than simply demonstrating zero missing words on a known vocabulary.

It would directly test the architectural proposition:

DIG does not need to know every complete word in advance in order to represent it.

09.5 Scaling With Word Length and Morphological Complexity

The present observations already indicate that vocabulary composition can affect runtime and energy consumption. Future tests should therefore avoid treating all words as computationally equivalent.

A systematic benchmark should stratify the dataset according to characteristics such as:

- number of characters,
- number of morphemes,
- number of compound constituents,
- derivational depth,
- ambiguity,
- number of possible lexical readings,
- and number of activated inference paths.

This would make it possible to determine what actually determines DIG's computational complexity.

For example, if runtime scales primarily with the number of relevant activated paths rather than with the total number of Hypernodes in the system, this would be an important architectural result.

Such measurements would also help answer a major scalability question:

Does increasing the total knowledge represented by DIG necessarily increase the computational cost of processing an individual word, or can processing remain predominantly local to the relevant activated subgraph?

This question is fundamental to the long-term scalability of the architecture.

09.6 Replacing the Preliminary Unsupported - Analysis Metric

The present benchmark uses the term “unsupported analyses as a preliminary description for documentable incorrect readings. For future scientific evaluation, this category should be replaced or decomposed into more precise linguistic error classes.

A neural prediction, a deterministic but incorrect lexical inference, an invalid morphological feature and an additional plausible lexical interpretation do not represent the same type of error.

Future evaluations should therefore classify errors according to their actual origin and linguistic severity.

Possible categories include:

- unsupported lemma,
- invalid morphological feature,
- invalid derivation,
- incorrect compound segmentation,
- unsupported lexical interpretation,
- incomplete representation,
- missing analysis,
- and valid alternative analysis not contained in the reference annotation.

This will make comparison between DIG and neural NLP systems considerably more informative.

The relevant question is not simply how many errors each system produces, but **what kind of errors each architecture produces and why**.

09.7 Normalizing Comparisons by Output Richness

Future benchmarks should also address an important limitation of direct runtime comparisons: the evaluated systems do not necessarily produce the same amount of linguistic information.

DIG is designed to return an interconnected lexical representation containing multiple possible dimensions, while another system may return only a lemma, part of speech and a limited set of morphological features.

A simple measurement of microseconds per input word therefore does not fully describe the computational work performed.

Future research should investigate additional normalized metrics, for example:

energy per valid linguistic attribute produced, **runtime per correct lexical interpretation**, or **computational cost per complete lexical representation**.

Such metrics would not replace conventional runtime and energy measurements, but they could make architectural comparisons considerably more meaningful.

If one system performs more linguistic construction while still requiring less computation, that distinction should be visible in the evaluation.

09.8 From Isolated Words to Sentences

The present benchmark deliberately isolates lexical processing in order to measure the current DIG layer without introducing additional contextual operations.

The next architectural stage must extend this representation into sentence-level inference.

This is where DIG's multi-representational structure may become particularly important. A word can possess several possible lexical interpretations in isolation. Context can subsequently activate, suppress or resolve these alternatives without requiring the original lexical information to be discarded.

This suggests a layered processing model:

lexical construction → multiple possible representations → contextual activation → sentence-level interpretation.

Such an architecture could allow ambiguity to remain explicitly represented until sufficient information exists to resolve it.

Future work should therefore evaluate how DIG performs:

- contextual lexical disambiguation,
- syntactic function assignment,
- relationships between words,
- agreement,
- semantic relations,
- and eventually larger sentence and discourse structures.

This represents a substantially more demanding research stage, but it is also where the distinction between DIG and a conventional morphological analyzer becomes increasingly clear.

A Appendix: 50-Word Test Set

The following table reproduces the 50-word test set and the manuscript's stated gold analyses and paradigm checkpoints. Its inclusion does not override the limitation described in the gold-standard status note above.

Table 11 50-word test set with the manuscript's stated gold analysis and paradigm checkpoints

#	WORD	CLASS	GOLD ANALYSIS	PARADIGM CHECKPOINTS
VERBS				
1	**bin**	Verb	sein: Ind.Pres.1.Sg	bin, bist, ist, sind, seid, war, wäre, sei, gewesen
2	**war**	Verb	sein: Ind.Pret.1.Sg + Ind.Pret.3.Sg	war, warst, waren, wart; Konj.II: wäre
3	**wäre**	Verb	sein: Konj.II.1.Sg + Konj.II.3.Sg	wäre, wärest/wärst, wären, wäret
4	**habe**	Verb	haben: Ind.Pres.1.Sg + Konj.I.1.Sg + Konj.I.3.Sg + Imp.Sg	hast, hat, hatte, hätte, gehabt
5	**werde**	Verb	werden: Ind.Pres.1.Sg + Konj.I.1.Sg + Konj.I.3.Sg + Imp.Sg	wirst, wird, wurde, würde, geworden
6	**wurde**	Verb	werden: Ind.Pret.1.Sg + Ind.Pret.3.Sg	wurde, wurdest, wurden, wurdet; Konj.II: würde
7	**kann**	Verb	können: Ind.Pres.1.Sg + Ind.Pres.3.Sg	kann, kannst, können, könnt; konnte, könnte, gekonnt
8	**dürfte**	Verb	dürfen: Konj.II.1.Sg + Konj.II.3.Sg	darf, darfst, dürfen; durfte, dürfte, gedurft
9	**sollte**	Verb	sollen: Ind.Pret.1.Sg + Ind.Pret.3.Sg + Konj.II.1.Sg + Konj.II.3.Sg	soll, sollst, sollen; sollte, gesollt
10	**ging**	Verb	gehen: Ind.Pret.1.Sg + Ind.Pret.3.Sg	gehe, gehst, geht, ging, gingst, gingen, ginge, gegangen
11	**läufst**	Verb	laufen: Ind.Pres.2.Sg	laufe, läufst, läuft, laufen; lief, liefe, gelaufen
12	**liest**	Verb	lesen: Ind.Pres.2.Sg + Ind.Pres.3.Sg	lese, liest, lesen; las, läse, gelesen
13	**aß**	Verb	essen: Ind.Pret.1.Sg + Ind.Pret.3.Sg	esse, isst, essen; aß, aßen, äße, gegessen
14	**nähme**	Verb	nehmen: Konj.II.1.Sg + Konj.II.3.Sg	nehme, nimmst, nimmt; nahm, nähme, genommen
15	**gibst**	Verb	geben: Ind.Pres.2.Sg	gebe, gibst, gibt, geben; gab, gäbe, gegeben
16	**schlägt**	Verb	schlagen: Ind.Pres.3.Sg	schlage, schlägst, schlägt; schlug, schlüge, geschlagen
17	**gedacht**	Verb	denken: Part.II	denke, denkst, dachte, dächte, gedacht
18	**brächte**	Verb	bringen: Konj.II.1.Sg + Konj.II.3.Sg	bringe, bringst, brachte, brächte, gebracht
19	**gesandt**	Verb	senden: Part.II, variant	sende, sendete/sandte, gesendet/gesandt
20	**gewesen**	Verb	sein: Part.II	sein, seiend, gewesen; bin/war/wäre
NOUNS				
21	**Mannes**	Noun	Mann: Gen.Sg	Mann, Mannes, Männer, Männern
22	**Männern**	Noun	Mann: Dat.Pl	Nom/Gen/Akk.Pl Männer; Dat.Pl Männern
23	**Kindes**	Noun	Kind: Gen.Sg	Kind, Kindes/Kinds, Kinder, Kindern
24	**Herzen**	Noun	Herz: Dat.Sg + Pl.all	Herz, Herzens, Herzen
25	**Namens**	Noun	Name: Gen.Sg	Name, Namens, Namen; Pl.all Namen
26	**Herrn**	Noun	Herr: Akk.Sg + Dat.Sg + Gen.Sg	Nom.Sg Herr; Pl.all Herren
27	**Studenten**	Noun	Student: Akk.Sg + Dat.Sg + Gen.Sg + Pl.all	Nom.Sg Student; otherwise Studenten
28	**Beamten**	Noun	Beamte(r): nominalized adjective inflection	Beamter/Beamte/Beamten/Beamtem depending on strong/weak/mixed
29	**Bücher**	Noun	Buch: Nom.Pl + Gen.Pl + Akk.Pl	Buch, Buches, Bücher, Büchern
30	**Lehrern**	Noun	Lehrer: Dat.Pl	Sg Lehrer/Lehrers; Nom/Gen/Akk.Pl Lehrer; Dat.Pl Lehrern
31	**Mädchens**	Noun	Mädchen: Gen.Sg	Sg/Pl Mädchen; Gen.Sg Mädchens
32	**Daten**	Noun	Datum: Pl.all	Datum, Datums; Pl.all Daten
33	**Kaktus**	Noun	Kaktus: Sg.all	Pl: Kakteen; variant: Kaktusse, Dat.Pl Kaktussen
34	**Status**	Noun	Status: Sg.all + Pl.all	all 8 case/number slots: Status

continued on next page

#	WORD	CLASS	GOLD ANALYSIS	PARADIGM CHECKPOINTS
ADJECTIVES / QUANTIFIERS				
35	**guten**	Adjective	gut: Pos + ADJ-EN	gut, besser, best-; am besten
36	**besserem**	Adjective	gut: Comp.Strong.Dat.Sg.M + Comp.Strong.Dat.Sg.N	besser, bessere, besserer, besserem, besseren
37	**besten**	Adjective	gut: Sup + ADJ-EN	best-, beste, bester, bestem, besten; am besten
38	**hohen**	Adjective	hoch: Pos + ADJ-EN, stem hoh-	hoch, hohe, hohen, höher, höchst-
39	**höher**	Adjective	hoch: Comp.uninflected, predicative/adverbial	höher, höhere, höherer, höherem, höheren
40	**nächsten**	Adjective	nah: Sup + ADJ-EN	nah, näher, nächst-; am nächsten
41	**vielen**	Adjective/ quantifier	viel: Pos + ADJ-EN	viel, viele, vielen, mehr, meist-
42	**mehr**	Adjective/ quantifier	viel: Comp.uninflected	viel → mehr → meist-
43	**teurer**	Adjective	teuer: Pos.Strong/Mixed -er slots + Comp.uninflected	teuer, teure, teurer, teuren, teuerst-
44	**dunklen**	Adjective	dunkel: Pos + ADJ-EN, stem dunkl-	dunkel, dunkle, dunklen, dunkler, dunkelst-
45	**edlen**	Adjective	edel: Pos + ADJ-EN, stem edl-	edel, edle, edlen, edler, edelst-
ARTICLES / PRONOUNS				
46	**der**	Article/ pronoun	DET: Nom.Sg.M + Gen.Sg.F + Dat.Sg.F + Gen.Pl; PRON: Nom.Sg.M + Dat.Sg.F	der, die, das, den, dem, des; pronominal also dessen/deren
47	**den**	Article/ pronoun	Akk.Sg.M + Dat.Pl	same slot value for article and pronoun
48	**deren**	Pronoun	Gen.Sg.F + Gen.Pl	relative/demonstrative pronoun, genitive form
49	**ihnen**	Pronoun	personal pronoun: 3.Pl.Dat	sie, ihrer, ihnen, sie
50	**welchem**	Article/ pronoun	Dat.Sg.M + Dat.Sg.N	welcher, welche, welches, welchem, welchen

BACK MATTER

Scientific References

[1] H. Schmid, A. Fitschen, and U. Heid. "SMOR: A German Computational Morphology Covering Derivation, Composition, and Inflection". In: *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*. Lisbon, Portugal: European Language Resources Association, 2004, pp. 1263–1266.

[2] H. Schmid. "A Programming Language for Finite State Transducers". In: *Finite-State Methods and Natural Language Processing*. Ed. by A. Yli-Jyrä, L. Karttunen, and J. Karhumäki. Vol. 4002. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 308–309. DOI: 10.1007/11780885_38.

[3] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. *spaCy: Industrial-strength Natural Language Processing in Python*. 2020. DOI: 10.5281/zenodo.1212303. URL: https://doi.org/10.5281/zenodo.1212303.

[4] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning. "Stanza: A Python Natural Language Processing Toolkit for Many Human Languages". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, July 2020, pp. 101–108. DOI: 10.18653/v1/2020.acl-demos.14. URL: https://aclanthology.org/2020.acl-demos.14/.

[5] Dudenredaktion, ed. *Duden – Die deutsche Rechtschreibung. Das umfassende Standardwerk auf der Grundlage der aktuellen amtlichen Regeln*. 29th ed. Vol. 1. Duden – Deutsche Sprache in 12 Bänden. Berlin: Dudenverlag, 2024, p. 1328. ISBN: 978-3-411-04019-3.

[6] W. Fleischer and I. Barz. *Wortbildung der deutschen Gegenwartssprache*. 4th ed. Berlin: De Gruyter, 2012. DOI: 10.1515/9783110256659.

[7] A. Nolda. *DWDSmor – German Morphology*. Software, version 0.18.0. Released 28 April 2026. 2026. URL: https://pypi.org/project/dwdsmor/.

[8] D. Altinok. "DEMorphy, German Language Morphological Analyzer". In: *CoRR* abs/1803.00902 (2018). DOI: 10.48550/arXiv.1803.00902. URL: https://arxiv.org/abs/1803.00902.

[9] F. A. Pirinen, S. N. Moshagen, and K. Hiovain-Asikainen. "GiellaLT — a Stable Infrastructure for Nordic Minority Languages and Beyond". In: *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*. Tórshavn, Faroe Islands: University of Tartu Library, May 2023, pp. 643–649. URL: https://aclanthology.org/2023.nodalida-1.63/.

[10] K. Lindén, M. Silfverberg, E. Axelson, S. Hardwick, and T. Pirinen. "HFST—Framework for Compiling and Applying Morphologies". In: *Systems and Frameworks for Computational Morphology*. Ed. by C. Mahlow and M. Piotrowski. Vol. 100. Communications in Computer and Information Science. Springer, 2011, pp. 67–85. DOI: 10.1007/978-3-642-23138-4_5.

[11] W. Klein and A. Geyken. "Das Digitale Wörterbuch der Deutschen Sprache (DWDS)". In: *Lexicographica* 26 (2010), pp. 79–96. DOI: 10.1515/9783110223231.1.79.

[12] S. N. Moshagen, T. A. Pirinen, and T. Trosterud. "Building an Open-Source Development Infrastructure for Language Technology Projects". In: *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*. Oslo, Norway: Linköping University Electronic Press, May 2013, pp. 343–352. URL: https://aclanthology.org/W13-5631/.

[13] A. Wöllstein and Dudenredaktion, eds. *Duden – Die Grammatik: Struktur und Verwendung der deutschen Sprache. Sätze – Wortgruppen – Wörter*. 10., völlig neu verfasste Auflage. Berlin: Dudenverlag, 2022. ISBN: 978-3-411-04050-6.

[14] E. Strubell, A. Ganesh, and A. McCallum. "Energy and Policy Considerations for Deep Learning in NLP". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3645–3650. DOI: 10.18653/v1/P19-1355. URL: https://aclanthology.org/P19-1355/.

[15] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau. "Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning". In: *Journal of Machine Learning Research* 21.248 (2020), pp. 1–43. URL: https://jmlr.org/papers/v21/20-312.html.

[16] X. Zhou, Z. Chen, X. Jin, and W. Y. Wang. "HULK: An Energy Efficiency Benchmark Platform for Responsible Natural Language Processing". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, Apr. 2021, pp. 329–336. DOI: 10.18653/v1/2021.eacl-demos.39. URL: https://aclanthology.org/2021.eacl-demos.39/.

[17] S. Poddar, P. Koley, J. Misra, N. Ganguly, and S. Ghosh. "Towards Sustainable NLP: Insights from Benchmarking Inference Energy in Large Language Models". In: *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational*

Linguistics: Human Language Technologies (Volume 1: Long Papers). Albuquerque, New Mexico: Association for Computational Linguistics, Apr. 2025, pp. 12688–12704. DOI: 10.18653/v1/2025.naacl-long.632. URL: https://aclanthology.org/2025.naacl-long.632/.

[18] Python Software Foundation. *Python 3.13 Documentation: time — Time Access and Conversions*. Documentation for time.perf_counter(). Python Software Foundation. 2026. URL: https://docs.python.org/3.13/library/time.html.

[19] Shelly Group. *Shelly Pro 2PM Documentation*. Device documentation for integrated power and energy metering. Shelly Group. 2026. URL: https://www.shelly.com/products/shelly-pro-2pm.

[20] NVIDIA Corporation. *NVML API Reference Guide*. NVIDIA Management Library documentation; includes GPU power and total energy queries. NVIDIA Corporation. 2026. URL: https://docs.nvidia.com/deploy/nvml-api/.

[21] Linux Kernel Documentation Project. *Power Capping Framework*. Documentation of the sysfs powercap interface and energy counters. The Linux Kernel. 2026. URL: https://www.kernel.org/doc/html/latest/power/powercap/powercap.html.

[22] PyTorch Contributors. *PyTorch Documentation: CUDA Memory Management and max_memory_reserved*. PyTorch. 2026. URL: https://docs.pytorch.org/docs/stable/cuda.

[23] CuPy Developers. *CuPy 14.1 Memory Management Documentation*. Documentation of GPU device memory pools and memory statistics. CuPy. 2026. URL: https://docs.cupy.dev/en/v14.1.0/user_guide/memory.html.

PUBLICATION & LEGAL NOTICE

© 2026 CeLLAI / FLEXCO. All rights reserved.

DIG, CeLLAI FlexCo and associated names, marks and logos are proprietary identifiers of their respective owners. All third-party product names, software names, trademarks and registered trademarks referenced in this publication remain the property of their respective owners.

Benchmark results reported in this publication were obtained under the hardware, software, dataset and measurement conditions documented herein. Performance, energy consumption, memory use and coverage may differ under other configurations, software versions, datasets or measurement procedures.

Third-party systems are referenced solely for scientific comparison. Their inclusion does not imply affiliation with, sponsorship by, or endorsement of this publication or its authors.