Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 291–296
DOI: 10.15439/2025F5480
ISSN 2300-5963 ACSIS, Vol. 43

Simple Idea Discovery in a Minimalist LLM Architecture Implementation

Robert Chihaia[1,2], Maria Trocan[1], Florin Leon[2]
[1]LISITE Lab, Isep, Paris, France, 28, rue Notre-Dame-des-Champs
75006 Paris, France, E-mail: maria.trocan@isep.fr
[2]Faculty of Automatic Control and Computer Engineering
"Gheorghe Asachi" Technical University of Iaşi, Bd. Mangeron 27, 700050 Iaşi, Romania
robert.chihaia@student.tuiasi.ro, florin.leon@academic.tuiasi.ro

***Abstract*—Large Language Models (LLMs) capture linguistic structure by operating on sequences of sub-word tokens, yet they often display behaviors that suggest an implicit grasp of high-level concepts. This study probes whether such "ideas" are genuinely encoded in LLM representations and, if so, how faithfully and to what extent. We created a deliberately minimalist LLM (encompassing both tokenizer and transformer architecture)designed to expose internal mechanisms with minimal architectural obscurity. Using a carefully curated toy corpora and probing tasks, we trace how semantically related prompts map onto the model's hidden states. Our findings reveal emergent clustering of conceptually similar inputs even in the stripped-down model. These insights advance our understanding of the representational geometry underpinning modern language models and outline a reproducible framework for future mechanistic studies of semantic abstraction.**

***Index Terms*—byte-pair encoding, tokenizer, minimalist LLM, sentiment spectrum, idea discovery**

I. Introduction

NATURAL Language Understanding (NLU) in contemporary LLMs is mediated by sub-word tokenization, a mechanism that excels at surface-level pattern matching but offers only an indirect handle on the abstract ideas that underpin human reasoning. Our research focuses on creating a new model class, whose computations are organized around explicit high-level abstraction rather than tokens. Motivated by neuro-cognitive evidence that human thought is concept-centric, we seek to inherit the efficiency and compositional complexity of the human brain's representation strategy while remaining compatible with modern neural training pipelines. As one of the first steps, we aim at replicating on a minimal scale a LLM architecture and probe it with a closely related variety of ideas. This experimental framework should provide a regime in which the internal representation of such models can be easily visualized.

Interpreting how LLMs encode "ideas" is difficult based on the sheer scale of their parameter spaces and the resulting high-dimensional activation patterns. Exhaustive inspection is computationally intractable, so similar work resorts to surrogate techniques, most commonly activation clustering and dimensionality reduction projections, to peek into the model's internal structure. While useful, these tools expose only narrow cross-sections of the computation, forcing mechanistic studies to test pre-selected hypotheses instead of capturing a holistic view of the network. This study follows this established framework, with the goal of capturing some form of patterns between abstract ideas withing the model's internal representation.

We have implemented a scall-scale LLM (around 31 million parameters), designed to test whether conceptual structures can be isolated and examined without the computational overhead of a large-scale LLM (tens of even hundreds of billions of parameters). The architecture involves a custom byte pair encoding tokenizer of 10256 vocabulary size to constrain the the embedding table and uses a standard Transformer decoder stack trained auto-regressively on the FineWeb corpus. To probe its internal representation, we create a set of sentences that differ only by sentiment adjectives spanning

$$negative \rightarrow neutral \rightarrow positive$$

positive continuum. Our hypothesis aims at activations which depict some form of smooth varying trajectories that would mimic the before mentioned continuum, allowing quantitative estimates of concept-correlation. The resulting architecture and probing framework showcases a lightweight and reproducible mechanistic study of idea representation in LLMs.

The paper is organized as follows: Section II introduces the methodology and the implemented tokenizer on a simple LLM architecture. Next, in Section III, we describe our experimental setup for probing sentiment. The results are presented and in Section IV, while Section V discusses some of the implications of our findings. Lastly, our conclusions are given in Section VI.

II. Methodology

A. Dataset

FineWeb is an openly released, web-scale corpus that Hugging Face and Forward assembled from ninety-six Common Crawl snapshots spanning from 2013 to 2024 [1]. After extraction the collection contains a little over 15 trillion tokens of English text, making it one of the largest high-quality resources currently available for pre-training language models. The pipeline first blocks known malicious or NSFW domains and applies sub-word heuristics to screen out undesirable URLs. Raw HTML that passes this gate is parsed with

Trafilatura to isolate the main article body. A FastText-based language detector then retains only documents whose English confidence exceeds 0.65. Quality is further improved with the repetition and formatting heuristics introduced in DeepMind's Gopher work, almost all the C4 filters, and several bespoke rules that discard list-like layouts or heavily duplicated lines. Each Common Crawl dump is then deduplicated independently using a 5-gram MinHash scheme (fourteen bands of eight hashes) before emails and public IPv4 addresses are anonymised. The full dataset is distributed under the ODC-BY 1.0 licence together with the exact code needed to rerun the pipeline, plus smaller random samples of 350 B, 100 B and 10 B tokens for users who want to prototype without downloading tens of terabytes. The current implementation uses approximately 11 billion tokens to train both the tokenizer and the LLM, in order to optimize alignment between the two components.

B. Tokenizer implementation

In [2], Byte-Pair Encoding (BPE) was repurposed from data compression to sub-word segmentation, laying the groundwork for all modern transformer tokenizers. Our implementation follows their recipe closely, but embeds the core merging algorithm directly into the tokenizer's training and inference routines.

We begin by applying the same regular-expression-based splitter adapted from OpenAI's tiktokenizer codebase (see `openai/tiktoken/tiktoken_ext/openai_public.py` on GitHub). This splitter segments text into "chunks" (words, punctuation marks, and whitespace tokens) ensuring that subsequent processing always respects linguistic boundaries. Each chunk is then encoded as a sequence of UTF-8 bytes, treating every byte as an atomic symbol and appending a special end-of-word marker to preserve chunk integrity. Once the entire corpus is reduced to byte-level symbol sequences, we calculate the frequency of every adjacent symbol pair across all chunks. In a word like `train_` for instance, the pairs `(t, r)`, `(r, a)`, `(a, i)`, `(i, n)`, and `(n, _)` each accumulate counts. At each of 10,000 iterations, the tokenizer locates the most frequent pair, merges it into a new minted token, substitutes every occurrence in the training data, and records this operation in a merge table. By the end of this process, our vocabulary comprises the original 256 byte symbols plus 10,000 newly minted sub-word tokens, for a total of 10,256 entries.

When encoding unseen text, the tokenizer replicates these same steps in reverse order but without rescanning the entire corpus. The input is normalized and chunked identically, then each chunk's byte IDs are greedily merged: the tokenizer repeatedly scans all adjacent ID pairs, consults the precomputed merge table and applies the earliest-created merge whenever possible. This process continues until no further merges remain, yielding a compact sequence of sub-word IDs that exactly mirror those learned during training.

By weaving the merge logic into both the training loop and the encoding routine, our implementation remains faithful to [2] original algorithmic design while providing a self-contained, efficient tokenizer.

C. LLM architecture

The current implementation is built on the decoder-only half of the Transformer architecture first introduced in [3]. By discarding the encoder and keeping only the auto-regressively masked decoder stack, the network learns to predict the next token in a sequence while only attending to previously seen tokens. [4] demonstrated that this seemingly small modification was enough to unlock impressive results based on the architecture's unsupervised capabilities when the model is trained at scale and exposed to a large corpus of text. The present code-base follows that blueprint: a single stream of tokens, augmented with learned positional vectors, is passed through a succession of identical decoder blocks, the final hidden states are projected back into the same embedding matrix to score the vocabulary right after one final layer normalization layer, and the entire network is optimized to minimize next-token cross-entropy.

Each decoder block in this implementation begins with a layer normalization applied to the incoming residual stream. This layer is followed by causal multi-head self-attention, implemented with FlashAttention introduced in [5], used to reduce the memory footprint while preserving the exact dot-product computation. After the attention output is added back to the residual stream, a second layer normalization prepares the signal for a position-wise feed-forward network whose hidden width is quadruple the model dimension and using the Gaussian Error Linear Unit (GELU) activation function. Another residual addition completes the block. The paired attention and feed-forward sub-layers serve complementary roles: attention lets every token selectively integrate information from earlier positions, capturing syntactic and semantic dependencies that may span multiple tokens, whereas the feed-forward network performs local feature transformations that help the model build richer hierarchical representations. Residual connections preserve gradient flow so that deeper stacks (eight in this implementation) can be trained without vanishing or exploding gradients, and the pre-norm ordering has proved more stable than the original post-norm scheme when the depth or learning rate is increased.

This configuration specifies a eight-block Transformer with eight attention heads operating in a 512-dimensional embedding space, a 512-token context window, and a 10 256-token vocabulary. These choices bring the total parameter count to 30.7 million parameters. For overall faithful replication, [6] and [4] have been used in conjunction for architecture design and hyper-parameters choices.

III. Experimental setup

A. Training process

The training pipeline that accompanies this implementation begins with a fully pre-tokenized dataset stored in memory. The tokens were produced off-line with the same byte-level

BPE vocabulary used by the model, so no run-time tokenization overhead is incurred. At start-up the loader reads only the array header, then maps the data lazily so that the text can fit into the process's virtual address space without exhausting RAM. A custom data loader performs an 80 / 20 split: the first eighty percent of the token stream is reserved for training, while the remainder for validation. Within each split it pre-computes non-overlapping blocks of tokens randomly, being fetched without substitution, i.e. until the entire training dataset has been completely traversed the already selected token cannot be part of a newly create batch.

In order to replicate the training procedure described in [6], the effective batch size is scaled up through gradient accumulation rather than through data parallelism. The script targets a global token batch of 524,288. Given the 8,192 tokens contained in a micro-batch (16 × 512), the loop accumulates gradients across 64 steps before issuing an optimizer step. This strategy keeps GPU memory usage roughly constant while preserving the training dynamics associated with large-batch optimization.

On the optimization front, AdamW with β_1 = 0.9, β_2 = 0.95 and decoupled weight decay provides the same training dynamics that have become standard in Transformer language models, while the initialization uses a depth-scaled normal distribution ($\sigma \approx \frac{0.02}{\sqrt{\text{layers}}}$) to keep activation variances well behaved from the first iteration.

Learning-rate scheduling follows the linear warm up and cosine decay template described in [6]. Over the first 715 steps the rate rises linearly from zero to its peak of 6×10^{-4}. Thereafter it decays following half a cosine toward a floor at ten percent of the maximum, reaching that floor at the final update step, 20,980.

Each optimization cycle ends with gradient-norm clipping to 1.0. This proves enough to manage the largest spikes that appear during the initial stages of the training process.

B. Probing methodology

Probing how sentiment emerges inside the model requires an experiment built around a simple stimulus: nine sentences that differ only in the adjective. All nine stimuli share the fixed template "The team's performance was [adj]" while varying only the adjective.

$$\text{terrible} \rightarrow \text{bad} \rightarrow \text{poor} \rightarrow \text{average} \rightarrow \text{decent}$$
$$\rightarrow \text{good} \rightarrow \text{great} \rightarrow \text{excellent} \rightarrow \text{outstanding}.$$

We have chosen the adjectives based on the trained vocabulary so that each adjective would represent exactly one token. Because every other token and every position in the sentence remains constant, any systematic pattern we discover in the activations must be driven by the single word that encodes sentiment. The adjectives were chosen to march monotonically from extreme negative `terrible` through neutral `decent` to extreme positive `outstanding`, giving us a one-dimensional sentiment continuum that is easier to reason about quantitatively.

To capture the activations within the model, we placed forward hooks on carefully chosen components of every decoder block. Before the attention mechanism has split its combined query-key-value projection, a hook captures the raw QKV tensor. Another hook records the attention output itself, head by head, exposing whether particular heads fire only for strongly negative or strongly positive descriptors. Finally, hooks on the feed-forward sub-layers collect the activations immediately before and after the non-linear transformation, revealing how the sentiment signal is reshaped by the position-wise network. In order to run the capture pass each sentence is tokenised with exactly the same byte-level BPE vocabulary that served during pre-training. The model is switched to evaluation mode so that no gradients accumulate, and the sentence is fed forward once. Every hook silently detaches its tensor, moves it to the CPU, and drops it into an ordered dictionary keyed by the module's name. When the forward pass ends, the raw text, the token ids, and the activation dictionary are stored into an object, producing one file per sentence.

Once the activations for the nine "team-performance" sentences have been serialized, the final task is to determine whether they encode a consistent sentiment signal and, if so, where that signal emerges most cleanly in the network's depth. To that end the evaluation framework applies three complementary dimensionality reduction techniques: principal component analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE) and uniform manifold approximation (UMAP) to the tensors captured by the hooks. Although each method attacks the curse of dimensionality in its own way, together they provide complementary view of the representational geometry that would be difficult to analyze in the original dimensional space.

IV. Results

Probing the model with a deliberately simple stimulus offers an insightful view through which to analyze "sentiment" materialize inside the network. One view point of our experiments involved an analysis of how strong the attention output is for each of the sentiment token within its particular sentence across every block. By constructing a heatmap of the L2 norm of each of the 512 dimensions of the attention layer and doing so for each individual block, we obtain a 8x9 matrix into how the strength of the sentiment token shifts as sentiment goes from negative to positive. We can see the results of this visualization in Figure 1.

Early blocks (zero to two) most likely focus on mixing low-level feature extraction, strong sentiment specific signals start around layer four. By layers five and seven, the magnitude for `great`, `excellent` and `outstanding` clearly exceed those for `bad`, `poor` and `terrible`. Neutral adjectives like `average` and `decent` dip in the second layer in a behavior which seems to indicate that the model tries to sort out neutral and charged adjectives. The final layer really lights up the `great` to `outstanding` end of the spectrum, which might suggest that those attention-outputs are driving whatever

downstream head is classifying or generating sentiment. The steep rise we see from layer four to five will emerge as a recurrent turning point when the activation in the MLP layer will be clustered together.

The output of the last MLP (Multi-Layer Perceptron) layer within each block called "mlp_down" was ran through a number of dimensionality reduction technique, namely t-SNE, UMAP and PCA. At the first two layers the representations resemble a scatter of unrelated dots. The dimensionality-reduced t-SNE and UMAP plots show no discernible structure. PCA, which can be observed in Figure 2, by extracting a dominant axis that explains roughly a quarter of the variance cannot align it with the desired semantic scale. In this stage the hidden states are still dominated by surface features which can be anything from individual token information, position or the model's learned statistical priors. Sentiment, though implicitly present in the parameters, has not yet formed a clear direction. By the third layer a gentle stretching becomes visible. Some of the adjectives begin to form clusters, which although imperfect, begin to showcase the way in which the model interprets them after the training process. One particularly interesting pattern appears in PCA where `terrible`, `decent`, `outstanding` form a clear line based on the fact that they represent the center and both ends of our sentiment continuum. An important transformation occurs in layers four, five and six. Here the feed-forward networks, amplify whatever directional hints attention has supplied. PCA forms in layers 5 and 6 a visually clear line of sentiment, although this behavior is not replicated in layer 7 and `average` seems to be left outside of this line in both occasions. Nonetheless, PCA confirms that impression quantitatively: the first principal component now captures close to thirty percent of total variance and almost perfectly sorts the nine sentences in the intuitive order a human would impose. The final layer rather shows an interesting feature, it snaps the representation once the polarity of the sentiment has been decided and the model seems to no longer waste capacity on this specific internal variation.

This progression could illustrate several broader principles about idea formation in transformers. Concepts do not appear fully formed but emerge through a gradual alignment process, first hinted at in the initial layers, then sharpened in the middle blocks and finally insignificant with regards to the other processes the model attends to. The path is one of dimensional collapse: the network trades representational breadth for semantic depth, compressing many token-level details into a low-dimensional subspace that encodes to some degree an abstract property. Even in a model stripped down to eight layers, the machinery is powerful enough to carve out that subspace within five or six steps of computation.

V. Discussion

The evolution of a concept inside a Transformer can be tracked, layer by layer, as the network reshapes raw token statistics into increasingly abstract structure. Our 31 M-parameter model, probed with nine sentiment-bearing sentences that differ only by a single adjective, offers a concise case study. In the first two blocks the activations appear almost patternless. By the third block, sentiment slowly starts to begin forming and some terms closely related in their end of the spectrum begin to form clusters. Middle layers encompass a geometry that is sentiment-shaped to a large degree. The model is, in effect, compressing many token-level distinctions into a low-dimensional manifold in which on of the axis embodies polarity.

The clarity we gain by reducing a language model to a handful of layers and feeding it a set of varying sentences may come at the cost of realism. A compact transformer, such as the eight-block system explored earlier, operates under severe representational constraints. It is forced to recycle the same few dimensions to encode grammar, knowledge and much more. While this scarcity makes the emergence of a single semantic axis easy to observe, it also means the network lacks the expressive width to host several concepts simultaneously without interference. There is also the matter of compositional breadth. A minimal model that excels at ranking adjectives inside a fixed syntactic frame may still struggle to generalize that knowledge when the structure mutates (say, when sentiment is conveyed through metaphor, multi-clause reasoning, or cross-sentence contrast). Without ample depth and width, the transformer cannot allocate separate sub-circuits for these varied pathways. Instead, what might happen is that it reuses the same mechanisms, which leads to brittle performance outside the probe's narrow domain.

The lessons drawn from watching sentiment take shape to some extend inside a minimal transformer reverberate well beyond the confines of our toy experiment. They suggest, first, that interpretability research should embrace a telescopic strategy: zoom in until the phenomenon of interest is clearly visible, and secondly, that research in areas discussed by [7] could in fact benefit from the same strategy. Small models reveal the basis of an idea, the incremental alignment, the dimensional collapses and the competition for representational space. Once these are mapped, larger architectures can be dissected along the same joints, guided by probes tuned on their miniature counterparts. The experiment also warns against relying on a single visualization or metric. t-SNE and UMAP although more common, PCA confirmed with a higher accuracy that those clusters lay along a nearly straight axis. In future work, multiple reduction lenses should be treated not as interchangeable but as complementary instruments, each sensitive to a different property of the hidden space. Moreover, comparison with already established open-source models would provide an insight into how this behavior appears in larger LLMs.

VI. Conclusion

This work explored how abstract ideas, such as sentiment, can emerge in the internal representations of a minimalist LLM. By constructing a lightweight model and probing it with controlled stimuli, we identified patterns of semantic organization that gradually form across layers. While the simplicity of the model limits generalization, it offers a

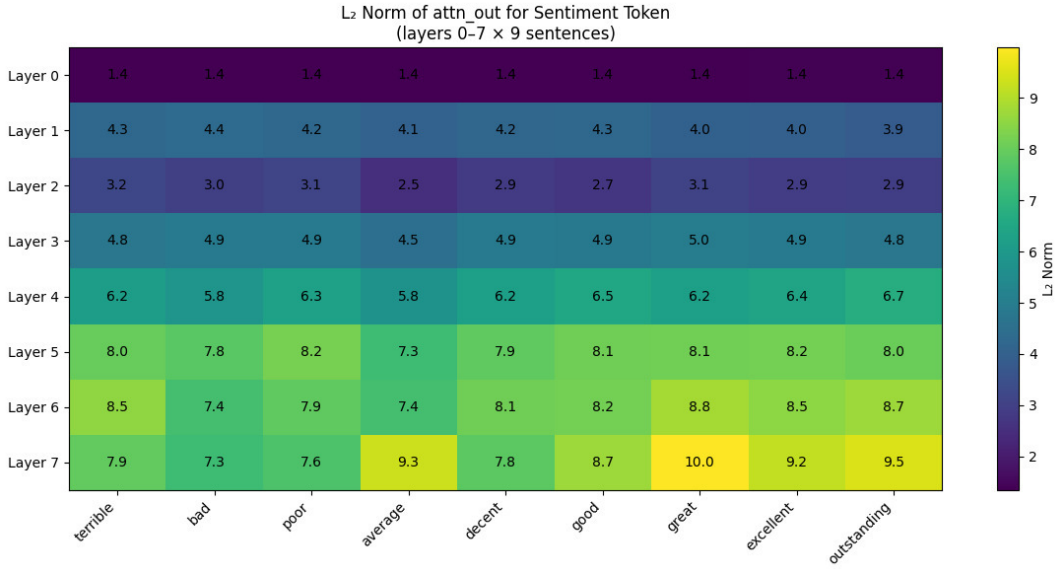

Fig. 1. Visualization of the attention heatmap

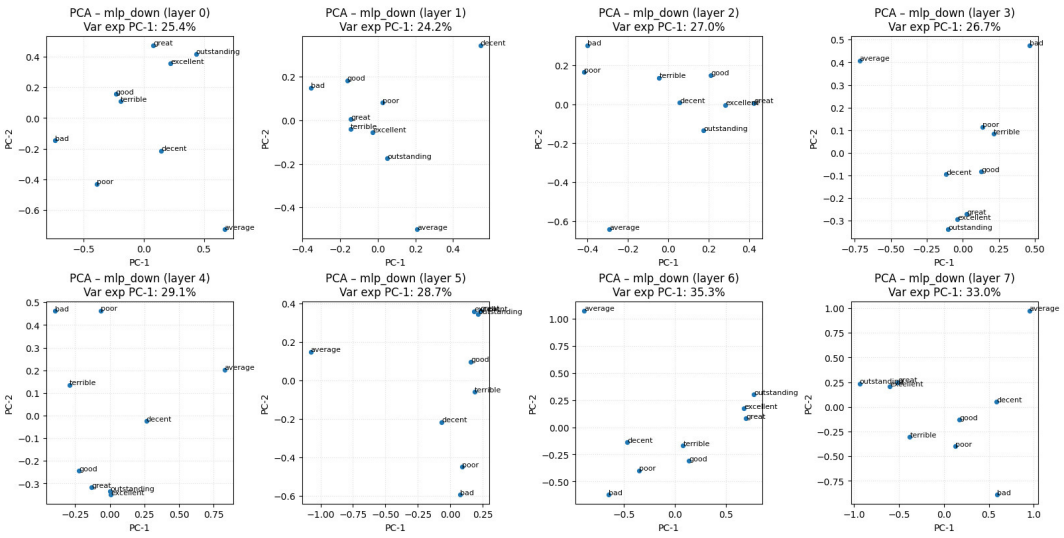

Fig. 2. Visualization of the MLP down projection layer in each layer using PCA

transparent platform for studying representational geometry. Our findings suggest that even constrained architectures can encode meaningful abstractions. They also motivate future comparisons with larger models to better understand how the representations of concepts change when the model size increases.

ACKNOWLEDGEMENT

This research is supported by the project "Romanian Hub for Artificial Intelligence - HRIA", Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 334906

REFERENCES

[1] G. Penedo, H. Kydlíček, L. B. allal, A. Lozhkov, M. Mitchell, C. Raffel, L. V. Werra, and T. Wolf, "The fineweb datasets: Decanting the web for the finest text data at scale," in *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. [Online]. Available: https://openreview.net/forum?id=n6SCkn2QaG

[2] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," 2015. [Online]. Available: https://arxiv.org/abs/1508.07909

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[5] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and

memory-efficient exact attention with io-awareness," *Advances in neural information processing systems*, vol. 35, pp. 16 344–16 359, 2022.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[7] L. Rolka, "Multi-criteria decision-making by approximation in the domain of linguistic values," in *Position Papers of the 18th Conference on Computer Science and Intelligence Systems*, ser. FedCSIS 2023, vol. 36. PTI, Oct. 2023. doi: 10.15439/2023f4728. ISSN 2300-5963 p. 97–102. [Online]. Available: http://dx.doi.org/10.15439/2023F4728