Efficient Patent Searching Using Graph Transformers

Krzysztof Daniell[1], Igor Buzhinsky[1] and Sebastian Björkqvist[1]

[1]IPRally Technologies Oy, Helsinki, Finland

Abstract
Finding relevant prior art is crucial when deciding whether to file a new patent application or invalidate an existing patent. However, searching for prior art is challenging due to the large number of patent documents and the need for nuanced comparisons to determine novelty. An accurate search engine is therefore invaluable for speeding up the process. We present a Graph Transformer-based dense retrieval method for patent searching where each invention is represented by a graph describing its features and their relationships. Our model processes these invention graphs and is trained using prior art citations from patent office examiners as relevance signals. Using graphs as input significantly improves the computational efficiency of processing long documents, while leveraging examiner citations allows the model to learn domain-specific similarities beyond simple text-based matching. The result is a search engine that emulates how professional patent examiners identify relevant documents. We compare our approach against publicly available text embedding models and show substantial improvements in both prior art retrieval quality and computational efficiency.

Keywords
patent search, prior art search, information retrieval, dense retrieval, graph neural networks

1. Introduction

Companies and individuals can protect their intellectual property by filing a patent application for their inventions. When granted, a patent provides its holder exclusive rights to the invention for a limited time. In exchange, the invention is publicly disclosed. The patent application process can be quite costly and time-consuming, so applicants typically seek a high certainty of it eventually being granted before initiating the application process. Since patents are granted only for novel inventions, conducting a prior art search is important to avoid unnecessary costs and delays. Finding relevant prior art is challenging due to the large number of existing patent documents and the detailed analysis required to distinguish novelty-destroying prior art from merely similar inventions. An effective patent search engine will both speed up the process and improve the results.

In this work, we present an approach for patent searching using graph representations of inventions as input to a Graph Transformer network. The graph of a patent document describes the core of the invention disclosed in that document, namely the features and the relationships between them. This representation condenses the original document, significantly reducing the computational resources required to process the document. The graphs are used as input to a Graph Transformer model trained on patent office examiner citations created during the application process for a patent. This enables the model to capture relationships between similar inventions, even when described using different terms. Furthermore, examiner citations help the model learn domain-specific terminology across different technical domains. The result is a patent search engine that efficiently retrieves relevant novelty-destroying prior art.
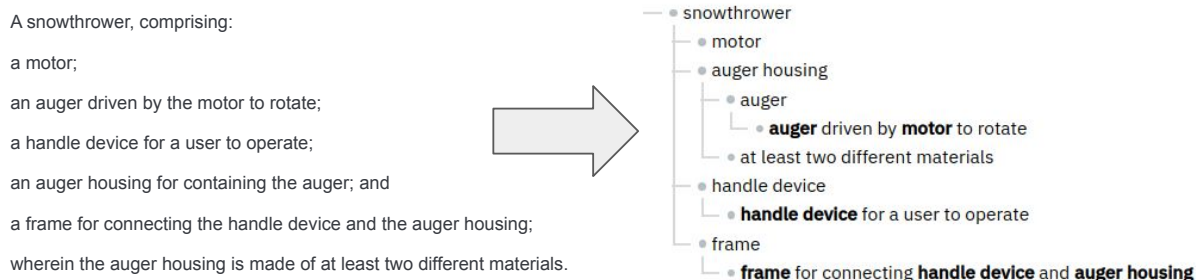
Figure 1: Example patent claim (US20170152638A1) converted into a graph.

2. Background

2.1. Patent Search Methods

Traditionally, patent searching has relied on Boolean search, where text-based matching is done using Boolean operators like AND, OR, and NOT [1]. Multiple Boolean search tools for patent searching exist, both paid [2, 3, 4] and free [5, 6, 7]. Performing high-quality Boolean searches often requires multiple iterations and substantial domain expertise to identify the most relevant terminology [8].

Machine learning methods have been explored for patent searching to address the limitations of Boolean searching. Traditional approaches including TF-IDF [9, 10] and word embeddings [11] have been applied. More recently, deep learning models such as BERT [12, 13, 14] and GPT-2 [15] have also been explored. In addition, several commercial machine learning-based patent search tools are available [16, 17, 18].

In our previous work [19], we describe a graph-based patent search engine where a Tree-LSTM model is trained for dense retrieval using patent office examiner citations as relevance signals. This paper extends that work by replacing the Tree-LSTM model with a Graph Transformer and improving the training procedure, as detailed in Section 3.

2.2. Graph Transformers

Graph Transformers adapt the Transformer architecture to graph-structured data [20]. Early models, such as Graphormer [21] and SAN [22], used full attention over all node pairs, which becomes computationally prohibitive as graph size grows—a critical concern for large patent corpora. Sparse Graph Transformers [23] address this by restricting attention to actual edges or small neighborhoods, drastically reducing overhead while preserving power for long-range dependencies. This is useful for patent searching where long documents revolving around a limited set of distinct features or relationships are typical. Mapping these concepts to nodes, with edges only where genuine conceptual links exist, yields a much sparser graph than a fully connected alternative.

2.3. Context-Based Representations

A recent approach in language modeling is using concept-level representations instead of token-based architectures. Large Concept Models [24] can be used to embed entire sentences—or "concepts"—into a high-dimensional, language-agnostic space rather than embedding every token separately. This perspective parallels our approach, where each key textual segment (e.g., a feature of the invention) is treated as a "concept node." We reduce computational overhead by applying sparse attention to these conceptual links yet still preserve the broad relationships essential for performing patent searches.

3. Approach

Our patent search engine consists of two main parts: We first convert each patent document into a graph that describes the invention, as described in Section 3.1. We then use a Graph Transformer model, described in Section 3.2, to embed the graph into a vector space for dense retrieval. The Graph Transformer model is trained for patent searching using patent examiner citations as relevance signals, as described in Sections 3.3 and 3.4.

The main differences to our previous work [19] are replacing the Tree-LSTM model with a Graph Transformer and enhancements to the training procedure.

3.1. Graphs

We convert each patent document into a graph to capture its core features and relationships while avoiding the overhead of full-text processing. An example of a graph is seen in Figure 1. Each node corresponds to a key feature of the invention (e.g., *snowthrower* or *motor*) or a text snippet indicating a relationship (e.g., *frame for connecting handle device and auger housing*). These relationships include hierarchical ones—such as part-of (meronym) or example-of (hyponym)—and functional ones, linking specific features to describe how they interoperate. By focusing on the essential technical structure, the graph becomes much smaller than the raw text while still conveying the essential features of the invention. This method aligns with how professional examiners perform novelty evaluations: they isolate core features and examine how they interrelate.

The details of how the graphs are created are described in our previous work [19]. In short, we first detect the features of the invention by doing a linguistic analysis with a natural language processing model. To find the relationships between the features, we use a set of hand-crafted rules designed to mirror the way patent professionals identify core inventive concepts and their interconnections from patent text. These rules detect terms describing relationships (e.g., comprising, connecting, containing) and use the output from the linguistic analysis phase to match the relationships with the correct features.

For each patent document, we create three different graphs: One containing only the first independent claim (later referred to as *first claim graph*), another containing all the claims (*all claims graph*), and one containing both the claims and the full description of the document (*description graph*).

3.2. Model Architecture

Our approach embeds each invention graph into a vector space for dense retrieval. Figure 2 illustrates four key steps, which we describe in detail below.

3.2.1. Node Embedding Initialization

Each token sequence (e.g. sentence or phrase) is tokenized using a BPE tokenizer [25] trained on patent documents. We pre-train the token embeddings using FastText [26] to improve the convergence speed of the model training. For each token sequence, we apply a Simple Word-Embedding-based Model (SWEM) [27], which computes mean and max pooled embeddings combined with a linear projection. This approach captures contextual information with a lower overhead than LSTMs or CNNs.

3.2.2. Graph Transformer Layers

Next, we feed the node embeddings into a Graph Transformer consisting of several layers that operate on the invention graph. We use Query-Key normalization [28] to stabilize training and reduce gradient variance. We also adopt a Pre-LayerNorm [29] architecture for consistent gradient flow. Each feed-forward sublayer uses Gated Linear Units with a GELU activation (GEGLU) [30]. We define adjacency according to the edges in the invention graph, ensuring that sparse attention focuses only on closely related nodes. This lowers computational costs while preserving crucial inter-node relationships [23].
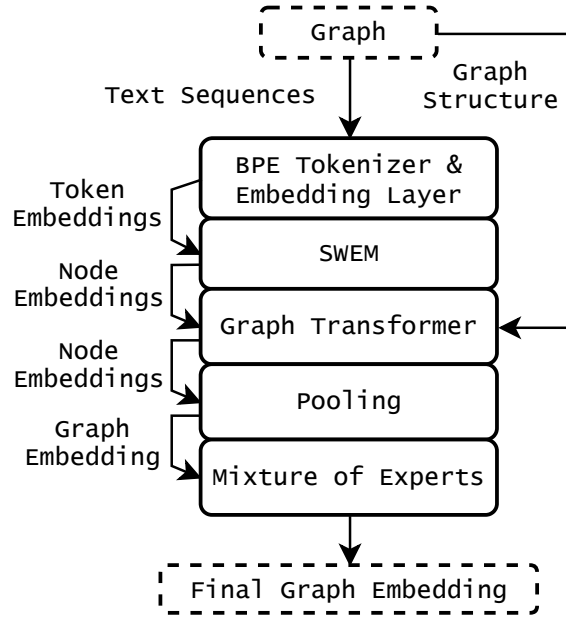
Figure 2: Proposed model architecture. We create initial node embeddings by using a BPE tokenizer and SWEM. The Graph Transformer layers refine these embeddings. We create a graph embedding as a weighted sum of all node embeddings and reduce the dimensionality of the graph embedding using a MoE layer.

3.2.3. Pooling

After the Graph Transformer layers refine each node representation, we assign each node a learned importance weight. We then compute a graph embedding using a weighted sum of all node embeddings, emphasizing the nodes most relevant to downstream tasks.

3.2.4. Dimensionality Reduction Layer

Lastly, we use a densely gated Mixture of Experts (MoE) [31] network to project the graph embedding to a lower dimension.

3.3. Training Data

The relevance signals used for training our model are citations extracted from patent examiner reports of patent applications. This approach ensures high-quality, expert-curated training data, as examiner citations highlight legally and technically relevant prior art—what we wish our search engine to place on top of search results.

We use the following citation categories: novelty-destroying prior art (labeled *X* by the EPO [32]), relevant prior art that does not invalidate novelty (*A citation*), and documents that show that the invention follows in an obvious way as a combination of existing inventions (*Y citation*). We include citations from more than 40 jurisdictions, with around 90% coming from US, EP, WO, JP, or CN. In total, we utilize approximately 31.7M citations formed from around 8.7M applications and 14.2M cited documents.

Each citation is represented by a pair of graphs, where the citing graph is typically the first claim graph, and the cited graph is the description graph. At training time, we use the following data augmentations and regularization techniques:

1. *Trivial citations:* We create artificial citations from the first claim graph and the description graph of the same patent. We observe that having such citations stabilizes the training.
2. *Graph type augmentation:* For each sample, with a predefined probability (0.4 in our experiments), we randomly replace either the citing graph or the cited graph (but not both) with the all claims

Table 1

Performance on retrieval of novelty-destroying citations. Higher is better for both recall and nDCG.

Approach	Seq. len.	Output dim.	Model size	Recall@3	nDCG@150
Dense retrieval models trained to embed patent documents					
Our, base stage	N/A	2048	156M	**0.4046**	**0.5564**
Our, dimensionality reduction stage	N/A	150	161M	0.3861	0.5372
Tree-LSTM, base stage	N/A	600	20M	0.3151	0.4685
PaECTER[1]	4×512	1024	345M	0.2798	0.4341
Dense retrieval models trained to embed text in general					
Stella[2]	3×2048	1024	435M	0.2734	0.4134
KaLM[3]	1×4096	896	494M	0.2211	0.3527
GTE-ModernBert[4]	1×4096	768	149M	0.2003	0.3231
Sparse retrieval models					
Okapi BM25 ($k_1 = 2.7, b = 1.15$)	N/A	N/A	N/A	0.1866	0.2874

graph. In addition, a replacement of the cited graph is forced for the easiest training samples according to one of our previous models.

3. *Node dropout:* We randomly drop some nodes from the graphs completely with probabilities that depend on the size of the graph.

4. *Embedding dropout:* Token embeddings retrieved from the embedding layer (Section 3.2.1) are subject to regular dropout.

Non-intersection between training, validation, and test sets is achieved on the document level, and no citations cross the different sets. When excluding a document from a set, we exclude all the documents of the same patent family (i.e., other applications corresponding to the same invention).

3.4. Training Procedure

We train our model using the PyTorch [33] and DGL [34] libraries. We employ triplet loss [35] as our loss function, assigning different margins to different citation categories. We use the AdamW [36] optimizer, reducing the learning rate by two on plateaus. Negatives are obtained using online hard negative mining [37] over the current batch. Batch creation accounts for graph sizes, making batch size dynamic, with 2100−2260 anchors and 900−960 positives on average. To make batches harder, we group samples into batches using International Patent Classification (IPC) [38] classes.

The training has two stages: A *base stage* with an output dimension of 2048, and a *dimensionality reduction stage* reducing the output dimension to 150 by fine-tuning the base stage and adding the MoE layer described in Section 3.2.4. While the base stage achieves higher recall, the dimensionality reduction stage strikes a better balance in terms of performance and vector storage cost.

The model is evaluated thrice per epoch. The training of each stage stops when the top-3 X citation recall (Recall@3, see Section 4.1) does not improve for three subsequent evaluation runs. When trained on eight L4 GPUs, the training process for both stages combined required approximately 185k updates (12 epochs) and took about 4.6 days.

---

[1]mpi-inno-comp/paecter

[2]NovaSearch/stella_en_400M_v5

[3]HIT-TMG/KaLM-embedding-multilingual-mini-v1

[4]Alibaba-NLP/gte-modernbert-base

4. Results

To evaluate the performance of our model in patent retrieval, we measure its effectiveness in retrieving novelty-destroying (X) citations of patent applications, as described in Section 4.1. The query is the first independent claim of the application, while we use the full text (all claims and description) for the search candidate documents. For our Graph Transformer model, we use the graphs corresponding to the query and search candidate documents as input, and the other approaches use the original texts. Any non-English documents are machine translated into English before being processed.

We compare our model with four text embedding models (Section 4.2), our previous Tree-LSTM-based approach (Section 4.3) and Okapi BM25 (Section 4.4).

4.1. Evaluation Procedure

To perform the evaluation, we use a test set containing about 161,000 search candidate documents and around 96,000 queries that cite one of the candidate documents as an X citation. We populate the search space by embedding each search candidate into a single vector, i.e., we do document retrieval, not passage retrieval. For each query we then perform a nearest neighbor search among all the search candidate documents and measure how often, on average, the document cited as X appears in the top three results (Recall@3). As an auxiliary metric, we use nDCG [39], which scores relevant document hits based on their positions. To compute nDCG, we use the top 150 search results (nDCG@150).

4.2. Comparison with Text Embedding Models

Based on MTEB leaderboard [40] scores, we selected three publicly available models to evaluate: Stella [41], KaLM [42] and GTE-ModernBert [43]. We selected these instead of larger models since they are of similar size as our model. We also compare our approach with PaECTER [12] as it is specifically trained using patent data.

We tuned the sequence length of these models to achieve the best Recall@3 on the validation set and then evaluated the models on the test set. To preserve more text, the two models that performed best on our evaluation sets (PaECTER and Stella) were applied on several chunks of the input text and averaged the embeddings.

PaECTER was trained on patent titles concatenated with abstracts as both queries and search candidates. However, we found that using the text of first claims as queries does not affect model performance, while using the full text (truncated to fit the 512 token window) for candidates even improves it. Thus, the only difference in inputs to PaECTER compared to other text embedding models is prepending input texts with patent titles.

Table 1 shows the chosen parameters of all models (resultant sequence lengths and the maximum number of input chunks, which is indicated by a digit before the "×" symbol), model sizes and the obtained metric values. The results show that our Graph Transformer-based approach outperforms the other models on novelty-destroying patent document retrieval. The recall of PaECTER is about 31% lower than our base stage model, while PaECTER performs about as well as Stella and better than the other text embedding models.

4.3. Comparison with Tree-LSTM

We also compare our approach with our previous approach based on Tree-LSTM [19], which was re-trained using the same training data used to train the Graph Transformer. Table 1 shows that its recall is 22% worse than our Graph Transformer base stage model, while outperforming PaECTER and all other evaluated text embedding models.

4.4. Comparison with BM25

BM25 [44] is a popular traditional word counting-based information retrieval approach that does not use machine learning. We tuned the hyperparameters of the Okapi BM25 version of this approach (k_1 and b) on 1,000 citations from our validation set (but preserving all available search candidates) and then computed its metrics on the test set. As seen in Table 1, the recall of BM25 is less than half of that of our base stage model and also trails all text embedding models we evaluated.

4.5. Limitations of Comparisons

Our comparison in Section 4.2 is limited since, apart from PaECTER, the other models were not trained specifically for patent retrieval. On the other hand, we found that using a large batch size is crucial for high recall when using online hard negative mining. If we would fine-tune text embedding models we couldn't use nearly as large batches as with our approach. With a sequence length of 512 and the GPU setup used for training our model, we could fit around 45-65 positives in a batch (depending on the model), which is more than 13 times less than we achieve with our Graph Transformer approach. This would also make the training process significantly slower and could impact the results due to the truncated texts.

We also note that this evaluation focuses on first-stage retrieval efficiency and effectiveness; comparison with computationally intensive re-ranking models like cross-encoders was considered beyond the scope of this specific study but represents a potential area for future investigation.

5. Conclusion

In this paper, we presented an approach for patent searching that utilizes a Graph Transformer model as a dense retriever. We compared the performance of our approach to existing text embedding models and showed that our approach achieves a significantly higher recall on novelty-destroying citation retrieval. Additionally, we demonstrated that our approach is more computationally efficient than existing text-based Transformer models.

6. Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check. And further ChatGPT-4o in order to: Paraphrase and reword, as well as improve writing style. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] N. S. Clarke, The basics of patent searching, World Patent Information 54 (2018) S4–S10. URL: https://www.sciencedirect.com/science/article/pii/S017221901630103X. doi:https://doi.org/10.1016/j.wpi.2017.02.006, best of Search Matters.

[2] Minesoft, Patbase, 2025. URL: https://minesoft.com/solutions/patent-intelligence/patbase/, [Online; accessed 22-Apr-2025].

[3] Patsnap, Patsnap, 2025. URL: https://www.patsnap.com, [Online; accessed 22-Apr-2025].

[4] Questel, Orbit intelligence, 2025. URL: https://www.questel.com/patent/ip-intelligence-software/orbit-intelligence/, [Online; accessed 22-Apr-2025].

[5] Google, Google patents, 2025. URL: https://patents.google.com/, [Online; accessed 22-Apr-2025].

[6] EPO, Espacenet, 2025. URL: https://worldwide.espacenet.com/, [Online; accessed 22-Apr-2025].

[7] USPTO, Patent public search, 2025. URL: https://www.uspto.gov/patents/search/patent-public-search, [Online; accessed 22-Apr-2025].

[8] A. Ali, A. Tufail, L. C. De Silva, P. E. Abas, Innovating patent retrieval: a comprehensive review of techniques, trends, and challenges in prior art searches, Applied System Innovation 7 (2024) 91.

[9] B. Herbert, G. Szarvas, I. Gurevych, Prior art search using international patent classification codes and all-claims-queries, in: C. Peters, G. M. Di Nunzio, M. Kurimo, T. Mandl, D. Mostefa, A. Peñas, G. Roda (Eds.), Multilingual Information Access Evaluation I. Text Retrieval Experiments, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 452–459.

[10] E. D'hondt, S. Verberne, CLEF-IP 2010: Prior art retrieval using the different sections in patent documents (2010).

[11] L. Helmers, F. Horn, F. Biegler, T. Oppermann, K.-R. Müller, Automating the search for a patent's prior art with a full text similarity search, PloS one 14 (2019) e0212103.

[12] M. Ghosh, S. Erhardt, M. E. Rose, E. Buunk, D. Harhoff, PaECTER: Patent-level representation learning using citation-informed transformers, 2024. URL: https://arxiv.org/abs/2402.19411. arXiv:2402.19411.

[13] U. U. Acikalin, M. Kutlu, Patent search using triplet networks based fine-tuned SciBERT, 2022. URL: https://arxiv.org/abs/2207.11497. arXiv:2207.11497.

[14] M. Freunek, A. Bodmer, Transformer-based patent novelty search by training claims to their own description, Applied Economics and Finance 8 (2021) 37. doi:10.11114/aef.v8i5.5182.

[15] J.-S. Lee, J. Hsiang, Prior art search and reranking for generated patent text, 2021. URL: https://arxiv.org/abs/2009.09132. arXiv:2009.09132.

[16] Ambercite, Ambercite, 2025. URL: https://www.ambercite.com/, [Online; accessed 22-Apr-2025].

[17] Amplified AI, Amplified, 2025. URL: https://www.amplified.ai/, [Online; accessed 22-Apr-2025].

[18] IPScreener, IPScreener, 2025. URL: https://ipscreener.com/, [Online; accessed 22-Apr-2025].

[19] S. Björkqvist, J. Kallio, Building a graph-based patent search engine, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 3300–3304.

[20] A. Shehzad, F. Xia, S. Abid, C. Peng, S. Yu, D. Zhang, K. Verspoor, Graph transformers: A survey, 2024. URL: https://arxiv.org/abs/2407.09777. arXiv:2407.09777.

[21] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, T.-Y. Liu, Do transformers really perform bad for graph representation?, 2021. URL: https://arxiv.org/abs/2106.05234. arXiv:2106.05234.

[22] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, P. Tossou, Rethinking graph transformers with spectral attention, 2021. URL: https://arxiv.org/abs/2106.03893. arXiv:2106.03893.

[23] V. P. Dwivedi, X. Bresson, A generalization of transformer networks to graphs, 2021. URL: https://arxiv.org/abs/2012.09699. arXiv:2012.09699.

[24] L. team, L. Barrault, P.-A. Duquenne, M. Elbayad, A. Kozhevnikov, B. Alastruey, P. Andrews, M. Coria, G. Couairon, M. R. Costa-jussà, D. Dale, H. Elsahar, K. Heffernan, J. M. Janeiro, T. Tran, C. Ropers, E. Sánchez, R. S. Roman, A. Mourachko, S. Saleem, H. Schwenk, Large concept models: Language modeling in a sentence representation space, 2024. URL: https://arxiv.org/abs/2412.08821. arXiv:2412.08821.

[25] R. Sennrich, B. Haddow, A. Birch, Neural machine translation of rare words with subword units, 2016. URL: https://arxiv.org/abs/1508.07909. arXiv:1508.07909.

[26] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146. URL: https://aclanthology.org/Q17-1010/. doi:10.1162/tacl_a_00051.

[27] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, L. Carin, Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms, in: I. Gurevych, Y. Miyao (Eds.), Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 440–450. URL: https://aclanthology.org/P18-1041/. doi:10.18653/v1/P18-1041.

[28] A. Henry, P. R. Dachapally, S. S. Pawar, Y. Chen, Query-key normalization for transformers, in: T. Cohn, Y. He, Y. Liu (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, Online, 2020, pp. 4246–4253. URL: https:

//aclanthology.org/2020.findings-emnlp.379/. doi:`10.18653/v1/2020.findings-emnlp.379`.

[29] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, T. Liu, On layer normalization in the transformer architecture, in: H. D. III, A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning, volume 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 10524–10533. URL: https://proceedings.mlr.press/v119/xiong20b.html.

[30] N. Shazeer, GLU variants improve transformer, 2020. URL: https://arxiv.org/abs/2002.05202. `arXiv:2002.05202`.

[31] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, Adaptive mixtures of local experts, Neural Computation 3 (1991) 79–87. doi:`10.1162/neco.1991.3.1.79`.

[32] EPO, Guidelines for Examination in the European Patent Office, 2024. URL: https://www.epo.org/en/legal/guidelines-epc/2024/b_x_9_2.html.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: an imperative style, high-performance deep learning library, Curran Associates Inc., Red Hook, NY, USA, 2019.

[34] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, Z. Zhang, Deep graph library: A graph-centric, highly-performant package for graph neural networks, 2020. URL: https://arxiv.org/abs/1909.01315. `arXiv:1909.01315`.

[35] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 815–823.

[36] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, 2019. URL: https://arxiv.org/abs/1711.05101. `arXiv:1711.05101`.

[37] A. Shrivastava, A. Gupta, R. Girshick, Training region-based object detectors with online hard example mining, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 761–769.

[38] World Intellectual Property Organization (WIPO), International Patent Classification (IPC), 2025. URL: https://www.wipo.int/classifications/ipc/en/, version 2025.01, [Online; accessed 22-Apr-2025].

[39] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of IR techniques, ACM Transactions on Information Systems (TOIS) 20 (2002) 422–446.

[40] M. Team, MTEB Leaderboard, https://huggingface.co/spaces/mteb/leaderboard, 2023. [Online; accessed 22-Apr-2025].

[41] D. Zhang, J. Li, Z. Zeng, F. Wang, Jasper and stella: distillation of sota embedding models, 2025. URL: https://arxiv.org/abs/2412.19048. `arXiv:2412.19048`.

[42] X. Hu, Z. Shan, X. Zhao, Z. Sun, Z. Liu, D. Li, S. Ye, X. Wei, Q. Chen, B. Hu, H. Wang, J. Yu, M. Zhang, Kalm-embedding: Superior training data brings a stronger embedding model, 2025. URL: https://arxiv.org/abs/2501.01028. `arXiv:2501.01028`.

[43] X. Zhang, Y. Zhang, D. Long, W. Xie, Z. Dai, J. Tang, H. Lin, B. Yang, P. Xie, F. Huang, M. Zhang, W. Li, M. Zhang, mGTE: Generalized long-context text representation and reranking models for multilingual text retrieval, in: F. Dernoncourt, D. Preoţiuc-Pietro, A. Shimorina (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, Association for Computational Linguistics, Miami, Florida, US, 2024, pp. 1393–1412. URL: https://aclanthology.org/2024.emnlp-industry.103/. doi:`10.18653/v1/2024.emnlp-industry.103`.

[44] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al., Okapi at TREC-3, Nist Special Publication Sp 109 (1995) 109.