

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

**EVALUATION OF VECTOR AND GRAPH-BASED SEARCH
METHODS IN A BANKING KNOWLEDGE PLATFORM
USING ADVANCED LANGUAGE MODELS**

M.Sc. THESIS

Bünyamin BAKIR

Department of Big Data and Business Analytics

Big Data and Business Analytics Programme

FEBRUARY 2025

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

**EVALUATION OF VECTOR AND GRAPH-BASED SEARCH
METHODS IN A BANKING KNOWLEDGE PLATFORM
USING ADVANCED LANGUAGE MODELS**

M.Sc. THESIS

**Bünyamin BAKIR
(528221083)**

Department of Big Data and Business Analytics

Big Data and Business Analytics Programme

Thesis Advisor: Assist. Prof. Dr. Süha TUNA

FEBRUARY 2025

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

**BANKACILIK BİLGİ PLATFORMU İÇİN VEKTÖR VE GRAFİK
TEMELLİ ARAMA YÖNTEMLERİNİN GELİŞMİŞ
DİL MODELLERİYLE DEĞERLENDİRİLMESİ**

YÜKSEK LİSANS TEZİ

**Bünyamin BAKIR
(528221083)**

Büyük Veri ve İş Analitiği Anabilim Dalı

Büyük Veri ve İş Analitiği Programı

Tez Danışmanı: Assist. Prof. Dr. Süha TUNA

ŞUBAT 2025

Bünyamin BAKIR, a M.Sc. student of ITU Graduate School student ID 528221083 successfully defended the thesis entitled “EVALUATION OF VECTOR AND GRAPH-BASED SEARCH METHODS IN A BANKING KNOWLEDGE PLATFORM USING ADVANCED LANGUAGE MODELS”, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Assist. Prof. Dr. Süha TUNA**
Istanbul Technical University

Jury Members : **Assoc. Prof. Dr. Ömer Faruk BEYCA**
Istanbul Technical University

Assist. Prof. Dr. Musa AYDIN
Fatih Sultan Mehmet Vakıf University

Date of Submission : **10 January 2025**
Date of Defense : **03 February 2025**





To my family,



FOREWORD

The journey of completing this Master's thesis has been both challenging and profoundly rewarding, representing not only a significant academic milestone but also personal growth. First of all, I sincerely thank my thesis advisor, Assist. Prof. Dr. Süha Tuna, whose expertise, insightful feedback, and unwavering support have been instrumental throughout this journey. His profound understanding of the field and dedication to academic excellence helped me overcome complex challenges and greatly enriched the quality of this work. I am deeply grateful to my beloved spouse, Feyza Tartar Bakır, whose endless patience, support, and belief in me were crucial during the most demanding phases of this investigation. Her encouragement provided the strength and determination needed to complete this thesis. Furthermore, I am sincerely grateful to my team, IT-Data Management, for their collaborative spirit and invaluable contributions throughout the research process. Their shared dedication and professional insight have greatly improved both the academic and practical aspects of this work.

FEBRUARY 2025

Bünyamin BAKIR

TABLE OF CONTENTS

	<u>Page</u>
FOREWORD	ix
TABLE OF CONTENTS	xi
ABBREVIATIONS	xiii
SYMBOLS	xv
LIST OF TABLES	xvii
LIST OF FIGURES	xix
SUMMARY	xxi
ÖZET	xxiii
1. INTRODUCTION	1
1.1 Purpose of Thesis	2
1.2 Literature Review	3
2. BACKGROUND	17
2.1 Large Language Models: An Overview of Language Modeling	17
2.1.1 Causal language modeling	19
2.1.1.1 Generative pretrained transformers	19
2.1.2 Masked language modeling	20
2.1.2.1 Bidirectional encoder representations from transformers	21
2.1.2.2 Robustly optimized BERT pretraining approach	22
2.1.2.3 Distillation bidirectional encoder representations from trans- formers	23
2.1.3 Translation language modeling	24
2.1.3.1 Text to text transfer transformer	24
2.1.4 Cross-lingual language model	26
2.1.4.1 Cross-lingual language model-roberta	27
2.1.4.2 Masked and permuted pretraining for language understanding ...	28
2.1.4.3 Mini language model	29
2.2 Text Embeddings	31
2.3 Word Embeddings	32
2.3.1 Traditional word embeddings	34
2.3.2 Contextualized embeddings	35
2.3.3 Universal text embeddings	36
2.3.3.1 Multilingual E5 text embeddings	36
2.4 Sentence Embeddings	37
2.4.1 Sentence BERT	37
2.4.2 Universal sentence encoder	39
2.5 Text Preprocessing	40
2.6 Similarity Search	41
2.6.1 Linear search	42
2.6.2 Radial basis function	42
2.6.3 Manhattan distance	42
2.6.4 Cosine similarity	43
2.6.5 Euclidean distance	44
2.7 Vector Indexing	45
2.7.1 The hierarchical navigable small world	45
2.7.2 Facebook AI similarity search	47
2.7.2.1 IndexFlatL2	48
2.7.2.2 IndexIVFFlat	48
2.7.2.3 Product quantization	49
2.7.2.4 Locality-sensitive hashing	50
2.8 Vector Search	50
2.9 Graph-Based Search	51
2.10 Text Summarization	52

3. METHODOLOGY	55
3.1 Data Preparation	55
3.2 Language Model Integration in Semantic Search: A Vector and Graph-Based Approach	56
3.3 Summarization	62
3.4 Performance Evaluation	63
3.4.1 Basic metrics	64
3.4.2 Performance metrics	66
3.4.3 Similarity metrics	70
3.4.4 Clustering metrics	73
3.4.5 Diversity metrics	76
3.4.6 Text generation metrics	80
3.4.7 Contextual metrics	88
3.4.8 Text quality metrics	92
3.4.9 Comprehension metrics	95
3.4.10 General language understanding evaluation	98
3.4.11 Evaluates with different index and similarity methods	100
4. CONCLUSIONS	105
REFERENCES	107
CURRICULUM VITAE	117



ABBREVIATIONS

AI	: Artificial intelligence
ANNS	: Approximate nearest neighbor search
BASE	: BERT base configuration
BGE	: Base Generative Encoder
BERT	: Bidirectional Encoder Representations from Transformers
CBOW	: Continuous Bag of Words
CLM	: Causal Language model
Word2Vec	: Word to Vector
CPU	: Central Processing Unit
DAN	: Deep Averaging Network
DBMSs	: Database management systems
DistilBERT	: Distillation BERT
E5	: Embedding for Everything
ELMo	: Embeddings from Language Models
F1	: Harmonic mean of precision and recall
FQ	: Focused Questions
GLUE	: General Language Understanding Evaluation
GPT	: Generative Pre-trained Transformer
HNSW	: Hierarchical Navigable Small World
k-NN	: k nearest neighbors
LaBSE	: Language-agnostic BERT Sentence Embedding
LDA	: Latent Dirichlet Allocation
LLMs	: Large language models
LM	: Language Model
LSH	: Locality-Sensitive Hashing
mBERT	: Multilingual BERT
mE5	: multilingual E5
MINILM	: Mini Language Model
MLM	: Masked Language Modeling
MLQA	: multilingual question answering benchmark
MPNET	: Masked and Permuted Pre-training
mT5	: Multilingual Pre-trained Text-to-Text Transformer
NER	: Named Entity Recognition
NLP	: natural language processing
NMF	: Nonnegative Matrix Factorization
NSP	: Next Sentence Prediction
PLM	: Permuted Language Modeling
PQ	: Product Quantization
RAG	: Retrieval-Augmented Generation
RBF	: Radial Basis Function
RNNs	: Recurrent Neural Networks

RoBERTa	: Robustly Optimized BERT Pretraining Approach
SBERT	: Sentence BERT
SQuAD	: Stanford Question Answering Dataset
STS	: Semantic Textual Similarity
T5	: Text-to-Text Transfer Transformer
TC	: Topic Coherence
TD	: Topic Diversity
TLM	: Translation Language Modeling
USE	: Universal Sentence Encoder
VDBMSs	: Vector DBMSs
XLM	: Cross-lingual Language Model
XLM-R	: Cross-lingual Language Model-Roberta
XLNet	: Extended Large Network Enhanced Transformer
XNLI	: cross-lingual natural language inference task
XQuAD	: Cross-lingual Question Answering Dataset
MRR	: Mean Reciprocal Rank
MAP	: Mean Average Precision
NDCG	: Normalized Discounted Cumulative Gain
BLEU	: Bilingual Evaluation Understudy
ROUGE	: Recall-Oriented Understudy for Gisting Evaluation
CIDEr	: Consensus-based Image Description Evaluation
NLU	: Natural Language Understanding
DBI	: Davies-Bouldin Index
ILS	: Intra-List Similarity
SST	: Stanford Sentiment Treebank
microsoft-m	: microsoft/Multilingual-MiniLM-L12-H384
all-mpnet	: all-mpnet-base-v2
paraphrase	: paraphrase-multilingual-MiniLM-L12-v2
distiluse	: distiluse-base-multilingual-cased-v2
fastText-t	: FastText Turkish
xlm-r-mult	: xlm-r-multilingual-v1
xlm-robert	: xlm-roberta-base
bge-large	: BGE-large-en-v1.5
mbert	: bert-base-multilingual-cased
IVF	: Inverted File
intfloat-m	: intfloat/multilingual-e5-large
CoLA	: Corpus of Linguistic Acceptability
RACE	: Recall, Accuracy, Consistency, and Efficiency
GloVe	: Global Vectors for Word Representation
FAISS	: Facebook AI Similarity Search
METEOR	: Metric for Evaluation of Translation with Explicit Ordering
MRPC	: Microsoft Research Paraphrase Corpus

SYMBOLS

D	: Vector
n	: Number of dimensions
p	: Probability distribution
τ	: Kendall's Tau
Q	: Set of queries
e	: Euler's number
exp	: Exponential
β	: Beta





LIST OF TABLES

	<u>Page</u>
Table 3.1 : GLUE Metrics Results.....	99





LIST OF FIGURES

	<u>Page</u>
Figure 2.1 : Types of language modeling Representations [1]	19
Figure 2.2 : BERT Input Representation [2]	22
Figure 2.3 : Illustration of Text-to-Text Framework [3]	25
Figure 2.4 : XLM Pretraining Architecture [4]	27
Figure 2.5 : Type Of Text Embeddings Models	31
Figure 2.6 : Different Eras of Text Embeddings [5]	32
Figure 2.7 : The King-Queen Relationship in 2D Embedding Space [6].....	33
Figure 2.8 : Illustration of the HNSW [7].....	46
Figure 2.9 : IndexFlatL2 computes the Euclidean (L2) distance by comparing a query vector against all points in the dataset. [8]	48
Figure 2.10 : The steps of PQ [9]	50
Figure 3.1 : Basic Metrics Results	65
Figure 3.2 : Performance Metrics Results	69
Figure 3.3 : Similarity Metrics Results.....	72
Figure 3.4 : Clustering Metrics Results	75
Figure 3.5 : Diversity Metrics Results	79
Figure 3.6 : Text Generation Metrics Results.....	86
Figure 3.7 : Contextual Metrics Results	91
Figure 3.8 : Text Quality Metrics Results.....	94
Figure 3.9 : Comprehension Metrics Results	97
Figure 3.10 : F1 Results	101
Figure 3.11 : Mean Reciprocal Rank Results	102
Figure 3.12 : Normalized Discounted Cumulative Gain Results	102
Figure 3.13 : Precision Results	103
Figure 3.14 : Recall Results.....	103



EVALUATION OF VECTOR AND GRAPH-BASED SEARCH METHODS IN A BANKING KNOWLEDGE PLATFORM USING ADVANCED LANGUAGE MODELS

SUMMARY

In the contemporary banking IT sector, the effective management and accessibility of technical documentation have emerged as crucial factors that directly influence organizational efficiency and knowledge sharing. In particular within large-scale financial institutions, timely access to accurate documentation is essential for the maintenance of complex systems, the sustainability of daily operations, and the swift resolution of technical issues. This research develops and evaluates an advanced semantic search system for technical documentation retrieval in the banking IT sector, implementing a comprehensive evaluation framework across multiple metric categories. The study analyzes twelve language models on a 3600 article knowledge-sharing platform, utilizing both vector-based and graph-based approaches.

Performance analysis across metric groups revealed distinctive patterns: Basic metrics demonstrated one of the Bidirectional Encoder Representations from Transformers (BERT) models, Language-Agnostic BERT Sentence Embedding (LaBSE) has superior performance (Harmonic mean of precision and recall (F1): 0.544, Precision: 0.6, Recall: 0.505), consistently outperforming other models by 15 – 20% across traditional retrieval metrics. Performance metrics showed vector search's efficiency (0.01 – 0.04s) versus graph search's consistency (40 + s), with vector approaches demonstrating up to 1000x faster response times. Similarity metrics highlighted vector search's higher Manhattan Distance scores (up to 20.7) while graph search maintained perfect rank correlation (1.0), indicating different strengths in semantic understanding.

Clustering metrics revealed graph search's superior Silhouette Scores (0.9), indicating better-defined clusters, with a 30% improvement over vector-based approaches. Diversity metrics showed notable variations, with the vector search achieving high embedding diversity (intfloat/multilingual-e5-large (intfloat-m) reaching 300) while the graph search maintained more normalized measurements. The implementation of different clustering approaches demonstrated the effectiveness of graph-based methods in maintaining semantic relationships between documents, with a 25% improvement in cluster cohesion.

Text Generation metrics indicated strong Bilingual Evaluation Understudy (BLEU) and Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scores for LaBSE (0.2 – 0.35), surpassing baseline models by 40%. Contextual metrics showed moderate performance in semantic coherence (0.11 – 0.14) but lower query relevance (0.02 – 0.03), identifying areas for future improvement. Text Quality metrics demonstrated high fluency (0.7) and readability (0.9) scores, showing a 50% improvement over traditional search approaches. Comprehension metrics revealed

areas for improvement in Recall, Accuracy, Consistency, and Efficiency (RACE) (0.2) and Stanford Question Answering Dataset F1 (0.05), though still outperforming baseline systems by 25%.

The results of the General Language Understanding Evaluation (GLUE) benchmark provided additional validation, with intfloat-m achieving the highest average score (0.7075), while specific models excelled in individual tasks (xlm-roberta-base (xlm-robert) in Corpus of Linguistic Acceptability (CoLA): 0.9562 and BGE-large-en-v1.5 (bge-large) in Semantic Textual Similarity (STS): (0.68).

The comprehensive analysis of similarity methods (Cosine, Euclidean, Manhattan, Radial Basis Function, Linear) revealed minimal performance variations in different metrics, with average performance differences remaining within the range 5 – 10%. Cosine similarity showed marginally better performance (3 – 5%) in technical document matching, while Manhattan distance proved more effective for numerical feature comparisons.

In contrast, indexing methods significantly impacted search quality and performance. Hierarchical Navigable Small World (HNSW) demonstrated superior results with 40 – 50% improvement in retrieval accuracy compared to basic indexing methods, while Flat_L2 showed 30 – 40% improvement. Across all evaluation metrics, these methods consistently outperformed other approaches like Inverted File (IVF)_flat, Product Quantization (PQ), and Locality Sensitive Hashing (LSH). IVF_flat showed variable performance, sometimes approaching Flat_L2's effectiveness but often falling short by 15 – 20%. PQ and LSH typically showed the weakest performance, with accuracy rates 25 – 35% lower than HNSW and a higher computational overhead (20 – 30% increase in processing time).

Resource utilization analysis revealed that HNSW achieved optimal performance with 15% less memory usage than Flat_L2 while maintaining comparable accuracy. The trade-off between index build time and search performance showed that HNSW requires 30% more initial setup time but delivers 45% faster search operations. This comprehensive evaluation establishes that indexing method selection is significantly more crucial than similarity metric choice for optimizing search performance in technical documentation systems. HNSW and Flat_L2 are the recommended approaches for similar implementations in banking IT knowledge management systems.

The findings provide concrete guidelines for implementing efficient search systems in technical environments, with quantifiable performance metrics across different operational scenarios. This study contributes to enhancing operational efficiency, speeding up problem-solving processes, and improving the use of institutional knowledge through better technical documentation management in the banking sector. It also offers valuable insights for documentation management in similar industries and provides practical solutions for optimizing technical knowledge management in large-scale financial institutions, laying a foundation for future research.

BANKACILIK BİLGİ PLATFORMU İÇİN VEKTÖR VE GRAFİK TEMELLİ ARAMA YÖNTEMLERİNİN GELİŞMİŞ DİL MODELLERİYLE DEĞERLENDİRİLMESİ

ÖZET

Günümüzde bankacılık IT sektöründe teknik dokümantasyonun etkin yönetimi ve erişilebilirliği, kurumsal verimliliği ve bilgi paylaşımını doğrudan etkileyen kritik bir faktör haline gelmiştir. Özellikle büyük ölçekli finans kuruluşlarında, karmaşık sistemlerin bakımı, günlük operasyonların sürdürülebilirliği ve teknik sorunların hızlı çözümü için doğru dokümana zamanında erişim hayati önem taşımaktadır. Artan dijitalleşme ve sistem karmaşıklığı ile birlikte, teknik dokümanların hacmi ve çeşitliliği de artmakta, bu durum geleneksel arama yöntemlerinin yetersiz kalmasına neden olmaktadır. Bankacılık sektöründeki teknik dokümanların çoğunlukla hassas ve kritik bilgiler içermesi, arama sistemlerinin güvenilirliğini ve doğruluğunu daha da önemli hale getirmektedir. Bu bağlamda geliştirilen araştırma, bankacılık IT sektöründeki teknik dokümantasyon yönetimini iyileştirmeyi hedefleyen gelişmiş bir semantik arama sisteminin geliştirilmesi ve değerlendirilmesine odaklanmaktadır.

Çalışmanın başlangıç fazında, farklı dil modellerinin teknik dokümantasyon aramalarındaki etkinliğini değerlendirmek için kapsamlı bir model seçim süreci yürütülmüştür. Seçilen 12 model arasında, farklı büyüklük ver türlerde geniş bir yelpaze yer almaktadır. Model seçiminde, çok dilli destek kapasitesi ve ölçeklenebilirlik kriterleri göz önünde bulundurulmuştur. Modellerin eğitim sürecinde, özellikle bankacılık sektörüne özgü teknik terimlerin ve kavramların doğru şekilde işlenmesi için özel bir önem verilmiştir.

Veri hazırlık aşamasında, JSON formatındaki HTML içerikli teknik dokümanlar üzerinde kapsamlı bir temizleme ve standardizasyon yapılmıştır. Toplam 3600 teknik makalenin bulunduğu bilgi paylaşım platformunda teknik ve önemli bilgiler korunarak temizleme işlemini yapılmasına özen gösterildi. Dokümanlar öncelikle format, yapı açısından normalize edilerek dil tespiti, karakter kodlaması düzeltmeleri, tekrarlayan girişlerin ayıklanması, tutarsız boşlukların kaldırılması ve eksik veya hatalı karakterlerin düzeltilmesi gibi temel ön işleme adımları uygulanmıştır.

Model eğitimi ve değerlendirme sürecinde, seçilen modeller farklı donanım konfigürasyonlarında test edilmiş, ölçeklenebilirlik ve kaynak kullanımı açısından detaylı performans analizleri gerçekleştirilmiştir. Her model için optimum batch size ve öğrenme oranı parametreleri deneysel olarak belirlenmiştir. Metin özetleme çıktıları, sonraki aşamalarda gerçekleştirilen benzerlik hesaplamaları ve kümeleme analizleri için temel girdi olarak kullanılmıştır.

Araştırma kapsamında uygulanan kapsamlı değerlendirme çerçevesi, on farklı metrik grubu üzerinden sistemin performansını analiz etmektedir. Temel metrikler değerlendirmesinde, LaBSE modeli kesinlik değerinde 0.7, duyarlılık

değerinde 0.7 ve F1 skorunda 0.7 ile üstün performans sergilemiştir. Graf tabanlı yaklaşımlar, özellikle distiluse-base-multilingual-cased-v2 (distiluse) ve paraphrase-multilingual-MiniLM-L12-v2 (paraphrase) modellerinde 0.5 – 0.6 aralığında tutarlı sonuçlar gösterirken, vektör tabanlı yaklaşımlarda all-mpnet-base-v2 (all-mpnet) ve paraphrase modelleri 0.4 – 0.5 aralığında orta düzey performans sergilemiş, xlm-robert ve microsoft/Multilingual-MiniLM-L12-H384 (microsoft-m) gibi modeller ise 0.1 civarında düşük performans göstermiştir.

Performans metrikleri açısından, vektör tabanlı yaklaşımlar 0.01 – 0.04 saniye aralığındaki yanıt süreleriyle, graf tabanlı yaklaşımların 40+ saniyelik işlem sürelerine göre belirgin bir üstünlük sağlamıştır. Yanıt süresi değerlendirmesinde FastText Turkish (fasttext-t) modeli vektör aramada, distiluse modeli ise graf aramada en hızlı yanıt sürelerini elde etmiştir. Ortalama kesinlik değerlendirmesinde LaBSE modeli 0.7 skoruyla öne çıkarken, graf tabanlı arama paraphrase (0.5) ve distiluse (0.4) modellerinde daha tutarlı performans göstermiştir.

Benzerlik metrikleri değerlendirmesinde, vektör aramanın intfloat-m modeliyle 20.7'ye varan Manhattan mesafesi skorları elde ettiği, xlm-r-multilingual-v1 (xlm-r-mult) ve paraphrase modellerinin 1.2 – 2.1 aralığında orta düzey mesafeler gösterdiği gözlemlenmiştir. Graf aramada ise xlm-r-mult 0.16, LaBSE ve paraphrase 0.1 – 0.13 aralığında skorlar elde etmiştir. Sıra korelasyonu analizinde, vektör aramada all-mpnet mükemmel korelasyon (1.0) gösterirken, microsoft-m, intfloat-m ve bert-base-multilingual-cased (mbert) modelleri 0.9 düzeyinde güçlü korelasyonlar sergilemiştir. Graf tabanlı aramada tüm modeller mükemmel sıra korelasyonu (1.0) göstermiştir.

Kümeleme metrikleri analizinde, graf aramanın mbert modeliyle 0.9'luk silüet skoru elde ettiği ve bge-large modelinin 0.8 ile takip ettiği görülmüştür. Vektör aramada bge-large ve xlm-r-mult 0.8'lik skorlarla öne çıkmıştır. Küme kalitesi metriğinde t5-base 0.6, fasttext-t ve bge-large 0.5 skorlarla güçlü performans gösterirken, all-mpnet ve paraphrase gibi modeller 0.1 civarında düşük performans sergilemiştir.

Çeşitlilik metriklerinde, intfloat-m modeli vektör aramada liste içi benzerlik skoru 17, gömme çeşitliliğinde 300 ve benzersizlik skorunda 22 gibi yüksek değerler elde ederken, graf arama daha tutarlı ancak düşük skorlar göstermiştir. Normalize çeşitlilik değerleri vektör aramada 0.3 – 0.9, graf aramada 0.3 – 0.4 aralığında seyretmiştir.

Metin üretimi metriklerinde, LaBSE modeli BLEU skorunda 0.2 ve ROUGE metriklerinde 0.3 – 0.35 değerleriyle öne çıkmıştır. Bge-large ve intfloat-m modelleri ROUGE değerlendirmelerinde 0.2 – 0.3 aralığında başarılı performans göstermiştir. Karmaşıklık ölçümlerinde modeller 1.0 – 1.3 aralığında tutarlı skorlar elde etmiştir.

Bağlamsal metrikler değerlendirmesinde, sorgu ilgililiği için bge-large 0.02, anlamsal tutarlılıkta LaBSE ve fasttext-t 0.11 – 0.12 ve konu tutarlılığında ise LaBSE ve paraphrase 0.13 – 0.14 skorlarına ulaşmıştır. Referans hizalamasında bge-large ve LaBSE 0.03 civarında en yüksek skorları elde etmiştir.

Metin kalitesi metriklerinde, microsoft-m, LaBSE ve intfloat-m modelleri akıcılık değerlendirmesinde 0.7, paraphrase ve xlm-r-mult modelleri okunurluk değerlendirmesinde 0.9'a varan skorlar elde etmiştir. Bağlamsal ilgililik değerleri ise 0.1 – 0.15 aralığında kalırken, LaBSE 0.15 ile en yüksek skoru elde etmiştir.

Genel dil anlama deęerlendirmelerinde, intfloat-m modeli 0.7075 ortalama skorla en yksek performansı gstermiřtir. Xlm-robert dilsel kabul edilebilirlik korpusu metrięinde 0.9562 ve bge-large anlamsal metin benzerlięi 0.6800 skorları elde edilmiřtir. Fasttext-t modeli tm metriklerde 0.4 – 0.5 aralıęında tutarlı ancak orta dzey performans sergilemiřtir.

İndeksleme ve benzerlik yntemlerinin karřılařtırma analizi, arama sisteminin genel performansında indeksleme ynteminin ok daha belirleyici olduęunu ortaya koymuřtur. Cosine, euclidean, manhattan, rbf ve linear gibi farklı benzerlik metodları, aynı model ve indeksleme yntemi iinde neredeyse zdeř performans gsterirken, indeksleme ynteminin seimini sonularda nemli farklılıklar yaratmıřtır. HNSW ve Flat_L2 indeksleme yntemleri tutarlı řekilde stn performans gstermiř, IVF_flat karıřık sonular verirken, PQ ve LSH en dřk performansı sergilemiřtir. HNSW, Flat_L2'ye kıyasla %15 daha az bellek kullanımıyla benzer doęruluk oranları yakalamıř, %30 daha uzun bařlangı kurulum sresine raęmen arama operasyonlarında %45'e varan hız artıřı saęlamıřtır. Bu bulgular, benzer sistemlerin implementasyonunda benzerlik metrięi seiminden ziyade indeksleme yntemi seimine ncelik verilmesi gerektięini, zellikle HNSW ve Flat_L2'nin tercih edilmesi gerektięini gstermektedir. Ancak nihai seimde hesaplama verimlilięi ve eklenebilirlik gibi pratik faktrlerin de gz nnde bulundurulması nemlidir.

Bu kapsamlı arařtırmanın sonuları, bankacılık IT sektrndeki teknik dokmantasyon sistemlerinde arama performansının optimizasyonunda, indeksleme yntemi seiminin benzerlik metrięi seimine gre daha kritik olduęunu ortaya koymuřtur. Arařtırma bulguları, teknik ortamlarda verimli arama sistemlerinin geliřtirilmesi iin farklı operasyonel senaryolarda kullanılabilecek somut ve llebilir performans metrikleri sunmanın yanı sıra, gelecekteki benzer uygulamalar iin de deęerli bir referans kaynaęı oluřturmaktadır. Bu alıřma, bankacılık sektrnde teknik dokmantasyon ynetiminin iyileřtirilmesi yoluyla operasyonel verimlilięin artırılması, sorun zme srelerinin hızlandırılması ve kurumsal bilgi birikiminin daha etkin kullanılması konularında nemli katkılar saęlamakta, aynı zamanda benzer sektrlerdeki dokmantasyon ynetimi alıřmaları iin de yol gsterici bir kaynak nitelięi tařımaktadır. Arařtırmanın sunduęu metodoloji ve bulgular, zellikle byk lekli finansal kuruluřlarda teknik bilgi ynetimi srelerinin optimizasyonu iin pratik zmler sunmakta ve gelecekteki arařtırmalar iin saęlam bir temel oluřturmaktadır.



1. INTRODUCTION

The digital transformation in the banking industry has led to an exponential increase in technical documentation and knowledge-sharing requirements. As financial institutions embrace technological advancement, the ability to efficiently access and utilize technical information becomes crucial for maintaining operational excellence and fostering innovation. This need is particularly evident in information technology departments, where rapid access to accurate technical documentation directly impacts system reliability and development efficiency.

Modern banking IT environments generate vast amounts of technical documentation, ranging from system specifications and implementation guides to troubleshooting procedures and development standards. While these documents contain valuable institutional knowledge, their utility is severely limited by inadequate search and retrieval mechanisms. The existing search system in many platforms relies primarily on exact title matching, failing to leverage the semantic richness of technical documentation or understand the contextual relationships between different documents.

The purpose of this research is to develop and evaluate an advanced semantic search system for technical documentation retrieval, incorporating multiple language models and search methodologies to significantly improve information accessibility and user experience in knowledge-sharing platforms. The study focuses on a banking IT knowledge-sharing platform containing 3,600 technical articles, implementing sophisticated search methodologies that understand both technical context and banking-specific terminology.

Through comprehensive analysis of various search methodologies, including both vector-based and graph-based approaches, this study aims to establish a framework for implementing effective search solutions in banking IT knowledge management systems. By evaluating twelve different language models of varying architectures

and sizes, along with different indexing methods and similarity metrics, the research seeks to optimize technical documentation retrieval while considering both accuracy and computational efficiency.

The findings will contribute to establishing best practices for implementing semantic search in technical documentation systems while providing insights for future improvements in similar platforms. This framework will serve as a foundation for enhancing information accessibility and knowledge-sharing efficiency in the banking technology sector.

1.1 Purpose of Thesis

The purpose of this research is to develop and evaluate an advanced semantic search system for technical documentation retrieval. This system incorporates multiple language models and search methodologies to significantly improve information accessibility and user experience in knowledge-sharing platforms. This study aims to establish a comprehensive evaluation framework that analyzes the effectiveness of different approaches while providing a foundation for future enhancements in similar systems.

In a knowledge-sharing platform containing 3600 technical articles, users face significant challenges in accessing relevant information due to fundamental limitations in the current search functionality. The existing search system operates solely on exact title matches, lacking the capability to analyze article content, understand semantic relationships, or evaluate contextual relevance. This restrictive approach, coupled with the platform's growing repository of technical documentation, severely limits users' ability to discover relevant articles unless they know the exact title, effectively rendering valuable information inaccessible. The current implementation's lack of semantic understanding and limited search capabilities results in poor retrieval accuracy and inadequate ranking of search results, leading to user frustration and underutilization of the knowledge base.

The absence of advanced search features such as content-based searching, semantic matching, and relevance ranking significantly impairs the platform's effectiveness

as a knowledge-sharing tool. Furthermore, the system's inability to understand the contextual relationships between articles and process natural language queries results in missed opportunities for knowledge discovery and reduced platform utility. These limitations are particularly problematic given the technical nature of the content and the diverse ways users might search for specific information.

This research addresses these critical limitations by developing and evaluating comprehensive search methodologies that incorporate 12 different language models of varying sizes and architectures, from large-scale models to more compact solutions. The study implements both vector-based and graph-based search approaches, analyzing their performance across multiple dimensions including semantic understanding, retrieval accuracy, and computational efficiency. Through a detailed evaluation of indexing methods and similarity metrics, this research aims to enhance information retrieval accuracy, improve user experience, and establish a framework for optimizing technical documentation search systems.

1.2 Literature Review

In a study focusing on the comparative analysis of Retrieval-Augmented Generation (RAG) methods, Emekci and Hakan (2024) provide a detailed evaluation of various RAG methodologies. These include Stuff, Refine, Map Reduce, Map Re-rank, Query Step-Down, and Reciprocal RAG, each of which is examined comprehensively in the article. The research underscores that context quality is more crucial than similarity scores for enhancing response accuracy. While similarity-based approaches may slightly improve scores by eliminating less relevant contexts, they often fall short in providing the detailed understanding needed for complex queries, suggesting that merely optimizing for similarity may be insufficient in applications requiring high precision. The study revealed notable trade-offs between response accuracy, token usage, runtime efficiency, and hardware utilization, necessitating careful optimization to balance these competing factors effectively. For instance, while the Reciprocal RAG method achieved impressive similarity scores of up to 91%, it also resulted in increased token usage, which can be a drawback in scenarios where resource constraints are a concern. This finding underscores the need for a strategic approach

to resource management, particularly in high-volume applications, such as chatbots, where efficiency is paramount.

Additionally, the study addresses the issues posed by ambiguous queries, noting that methods designed to handle ambiguity produce higher similarity scores. The 'Step-Down' technique, for instance, proved effective in managing ambiguity, outperforming other methods that do not address this issue in terms of similarity scores. This highlights the importance of efficient document filtering and ranking to ensure the generative model receives relevant context, which is essential for producing accurate responses. In conclusion, this research emphasizes the importance of optimizing various parameters in RAG-based applications, including response accuracy, vector store scalability, and resource management. The grid-search optimization results revealed nuanced performances across different configurations, indicating that RAG methods can be tailored to meet the specific needs of various domains such as finance, insurance, and healthcare, where accurate and contextually relevant responses are critical. The findings advocate for a more refined approach to RAG optimization, aiming to enhance both the efficiency and effectiveness of language models in generating high-quality outputs [10].

A recent comprehensive review of vector databases by Yikun Han, Chunjiang Liu, and Pengfei Wang (2023) highlights that approximate nearest neighbor search (ANNS) techniques in vector databases produce diverse outcomes depending on their underlying principles and inherent advantages and disadvantages. Hash-based approaches are known for their speed and scalability, achieving fast query times owing to their compact binary representations of data. However, although they perform well with large datasets, their accuracy can be affected by hash collisions, particularly in high-dimensional spaces. In contrast, tree-based approaches provide efficient search capabilities by effectively narrowing down the search space, especially in lower dimensions.

They typically yield accurate results in identifying nearest neighbors for well-structured data; however, their performance can deteriorate significantly as dimensionality increases, resulting in longer query times and reduced precision. On

the other hand, graph-based approaches often achieve high recall rates, meaning they can effectively identify many of the nearest neighbors, particularly in complex datasets. Their flexibility allows them to adapt to various data distributions and relationships, thereby providing robust results across diverse scenarios. However, the complexity involved in graph construction and maintenance can lead to longer setup times and higher memory usage, which may affect overall performance. Finally, quantization-based approaches excel in storage efficiency, significantly reducing the memory required for high-dimensional data, which is particularly beneficial for large datasets. They can also enhance the query speed by simplifying the data representation, thereby enabling faster comparisons. Nevertheless, there is often a trade-off between speed and accuracy, as dimensionality reduction can result in information loss, potentially affecting the quality of the nearest neighbor results.

In summary, hash-based and quantization-based techniques generally provide the fastest query times, while tree-based methods may slow down in high dimensions, and graph-based methods can be slower due to their inherent complexity. Tree-based approaches typically offer high accuracy in lower dimensions, whereas hash-based methods may struggle with accuracy owing to collisions. Graph-based methods can provide high recall but may vary in precision depending on the graph structure, whereas quantization methods may sacrifice some accuracy for speed and storage efficiency. Ultimately, the selection of an ANNS methodology should be guided by the specific requirements of the application, including the nature of the data, dimensionality, and the desired balance between speed, accuracy, and scalability. A thorough understanding of the strengths and limitations of each approach is crucial for selecting the most suitable method for a given scenario [11].

The study by Xie, Xingrui, Liu, Han, Hou, Wenzhe, and Huang, Hongbin (2023) titled "A Brief Survey of Vector Databases" explores the crucial function of vector databases in handling and examining high-dimensional data, effectively tackling issues that conventional databases find challenging. As big data and artificial intelligence (AI) increasingly influence various sectors, the limitations of conventional databases, such as inadequate storage capacity and processing power, have become apparent. This study outlines the workflow of vector databases, which consists of two primary phases:

indexing and querying. During indexing, complex data are transformed into vector embeddings, which are subsequently compressed into representative vectors. This process ensures that similar data points are grouped, thereby facilitating an efficient retrieval. The querying phase involves transforming a user query into a vector and performing both rough and detailed comparisons to identify the most similar vector data.

The methodologies discussed in this paper focus on various similarity search algorithms and metrics that enhance the search efficiency in vector databases. Key algorithms include K-means, which partitions data into clusters to maximize intra-cluster similarity; LSH, which organizes similar data points into designated buckets for faster access; HNSW, which constructs a hierarchical graph structure to link similar data points; and PQ, which compresses high-dimensional vectors to improve similarity search performance. The paper also details similarity metrics such as Euclidean Distance, which measures the straight-line distance between vectors; Dot Product Similarity, which evaluates the similarity based on the dot product of vectors; and Cosine Similarity, which assesses the cosine of the angle between two vectors to determine their directional similarity.

The research compares several popular vector database products, including Faiss, pgvector, Chroma, Milvus, and Pinecone, emphasizing their capabilities in processing high-dimensional data and supporting machine learning tasks. This comparison underscores the need for systematic research into these databases' theoretical foundations and performance, as well as the importance of factors like availability, reliability, and compatibility when choosing a vector database. The survey concludes that vector databases are vital for efficiently retrieving approximate results from extensive high-dimensional datasets, utilizing advanced similarity search algorithms and metrics that significantly improve search efficiency and reduce computational overhead. The paper also proposes potential areas for future research in the evolving field of vector databases [12].

Another research conducted by Toni Taipalus (2024) provides a comprehensive overview of Vector database management systems (VDBMSs), delving into their

fundamental concepts, methodologies, use cases, and current challenges, offering valuable insights into the evolving landscape of vector data management. The research begins by highlighting the growing need to handle complex, unstructured data like text, images, and videos through vectorization, a process that transforms data into high-dimensional numeric vectors for more efficient processing and retrieval.

VDBMSs are specialized systems designed to manage vector data, enabling efficient storage, retrieval, and processing. Unlike traditional database management systems (DBMSs), which primarily focus on structured data, VDBMSs are tailored for high-dimensional vectors and support similarity searches based on various distance metrics such as Euclidean distance and cosine similarity. This study employs a narrative literature review to synthesize existing knowledge on VDBMSs and conducts a comparative analysis of different vector indexing techniques. It also develops a conceptual framework to clarify key terms and components, explore practical use cases, identify challenges, and discuss future trends in the field.

Several applications of VDBMSs are highlighted, including image similarity search, where the system finds images that are visually similar to a given input; recommender systems, which suggest products or content based on user preferences and behaviors; and chatbots, which utilize long-term memory to enhance user interactions by recalling past conversations. This study identifies several challenges faced by VDBMSs, such as high dimensionality, which makes managing and processing data with many dimensions computationally intensive; sparsity, where many vectors may contain numerous zero values, complicating storage and retrieval; and maturity and security, as VDBMSs are relatively new compared to traditional DBMSs and may lack robust security features and community support.

Regarding future trends, the researchers anticipate growing demand for VDBMSs, leading vendors to concentrate on developing new algorithms for vector indices and enhancing the human readability of high-dimensional vectors through visualizations. They also foresee advancements in incremental learning techniques to make model retraining more cost-effective. The study concludes by underscoring the importance of VDBMSs in the evolving data landscape, emphasizing their potential to facilitate

effective vector data management and addressing the challenges associated with high-dimensional data. Overall, it serves as a fundamental resource for comprehending the principles, methodologies, and applications of vector database management systems, offering insights into their current state and future directions [13].

The research by Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019) introduces BERT, a model designed to enhance natural language understanding by overcoming the limitations of traditional unidirectional language models. These models, which process text in a single direction (either left-to-right or right-to-left), often struggle with tasks that require a comprehensive understanding of the context. BERT overcomes this challenge by employing a bidirectional approach, which allows it to consider the entire context of a sentence simultaneously. The architecture of BERT is based on the transformer model, which utilizes self-attention mechanisms to effectively process input sequences.

BERT undergoes a two-step training process: pretraining and fine-tuning. During pre-training, BERT employs two main objectives: Masked Language Model (MLM) and Next Sentence Prediction (NSP). The MLM randomly masks a percentage of the input tokens and trains the model to predict these masked tokens based on their surrounding context, thereby learning rich bidirectional representations. The NSP task involves predicting whether one sentence logically follows another, which helps the model to understand the relationships between sentences. After pre-training, BERT can be fine-tuned on specific tasks with minimal additional parameters, making it versatile and efficient for various applications in natural language processing (NLP).

This study evaluates BERT's performance on the GLUE benchmark, which encompasses a variety of tasks such as sentiment analysis, paraphrase detection, and question answering. BERT's performance is compared that with of other state-of-the-art models, highlighting its effectiveness. One of the key advantages of BERT is its ability to capture nuanced meanings and relationships in a language through bidirectional contextualization. This capability is particularly beneficial for tasks, such as question answering and language inference. The authors emphasized the significance of transfer learning in NLP, where a model pre-trained on a large dataset

can be fine-tuned on smaller, task-specific datasets, leading to improved performance even in low-resource scenarios.

BERT achieved remarkable results on the GLUE benchmark, scoring 80.5%, which represents a 7.7% absolute improvement over the previous state-of-the-art. On the Stanford Question Answering Dataset (SQuAD) dataset, BERT reached a test F1 score of 93.2, marking a 1.5 point absolute improvement over the best existing model at the time. For SQuAD v2.0, it achieved a test F1 score of 83.1, reflecting a 5.1 point improvement. BERT demonstrated strong performance across a wide range of NLP tasks, including achieving an accuracy of 86.7% on the Multigenre Natural Language Inference dataset, which is a 4.6% absolute improvement.

In conclusion, this study posits that BERT represents a significant advancement in the field of NLP. Its bidirectional architecture, combined with effective pretraining and fine-tuning strategies, enables it to achieve state-of-the-art results across multiple tasks. BERT's design of BERT is conceptually simple and empirically powerful, paving the way for future research and applications in NLP. These findings underscore the importance of rich, unsupervised pre-training in developing effective language understanding systems, demonstrating BERT's potential of BERT to transform the landscape of NLP [14].

Another work by Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel (2021) presents a comprehensive overview of the Text-to-Text Transfer Transformer (T5) architecture, followed by its multilingual variant, the Multilingual Pre-trained T5 (mT5) model, which is designed to enhance cross-lingual understanding and performance. The primary goal of the mT5 model is to create a massively multilingual model that can perform a wide range of NLP tasks across multiple languages. The authors aimed to demonstrate that a simple pre-training recipe can yield competitive performance in cross-lingual representation learning without relying on complex techniques or extensive parallel data.

mT5 is built on transformer architecture, known for its effectiveness in handling sequential data, and employs a text-to-text framework, meaning that both inputs and

outputs are treated as text strings. This unification allows the model to handle various tasks such as translation, summarization, and question-answering using the same architecture. The model was pre-trained on the Multilingual Common Crawl dataset, a multilingual version of the Colossal Clean Crawled Corpus dataset, which included data from 101 languages. This diverse dataset is crucial for enabling the model to generalize across different languages and tasks, as it is sourced from a variety of web pages, ensuring a broad representation of language use.

The authors evaluate mT5 on several benchmarks, including a cross-lingual natural language inference (XNLI) task, an SQuAD comprehension benchmark, a multilingual question-answering benchmark (MLQA), and a multilingual question-answering dataset. The results indicate that mT5 achieves state-of-the-art performance on all classification and question-answering tasks, demonstrating strong zero-shot performance, meaning it can generalize to languages and tasks that it has not explicitly been trained on. The authors introduced techniques to mitigate "accidental translation," which can occur when the model generates outputs in a different language than intended.

In terms of comparative performance, mT5 demonstrated superior accuracy across various tasks compared to other existing multilingual models. For instance, in the XNLI task, Multilingual BERT (mBERT) achieved an accuracy of 65.4%, whereas XLM performed slightly better, at 69.1%. However, mT5 surpasses both, demonstrating its ability to generalize effectively across languages. In the Paraphrase Adversaries from Word Scrambling – Cross-lingual task, mBERT scored 81.9% accuracy, and XLM followed closely, with 80.9% accuracy. The mT5 model's performance in this area further emphasizes its robustness in handling paraphrase identification across languages.

When evaluating the models on structured question-answering tasks, mBERT achieved an F1 score of 62.2, whereas XLM scored 61.2. By contrast, mT5 outperformed both models, indicating its enhanced capability to understand and generate responses to questions in a multilingual context. For the wikipedia annotated Named Entity Recognition (NER) task, mBERT and XLM show F1 scores of 64.5 and 59.8,

respectively, whereas mT5's results highlight its effectiveness in NER, showcasing its ability to identify and classify entities accurately across different languages. The results on the Cross-lingual Question Answering Dataset and MLQA benchmarks further illustrate mT5's state-of-the-art performance, with F1 scores and exact match metrics exceeding those of mBERT and XLM.

Overall, the mT5 model represents a significant advancement in the field of multilingual NLP, demonstrating that a straightforward pretraining approach can yield high performance across a variety of tasks and languages. The comparative outputs of mT5 against other models underscore its significant advancements in multilingual NLP tasks, demonstrating that it not only matches but also often exceeds the performance of its predecessors. This positions mT5 as a leading model in the field capable of handling a diverse array of tasks with high accuracy and efficiency, thereby paving the way for future developments in multilingual representation learning. The authors emphasize the model's scalability and effectiveness, and they make the model checkpoints and code publicly available, promoting further research and application in multilingual NLP [15].

The article titled "Performance of 4 Pre-Trained Sentence Transformer Models in the Semantic Query of a Systematic Review Dataset on Peri-Implantitis" explores the effectiveness of various pre-trained sentence transformer models in analyzing a systematic review dataset focused on peri-implantitis. Systematic reviews play a crucial role in evidence-based medicine by consolidating research outcomes to guide clinical practices. However, the growing volume of literature poses challenges in efficiently identifying relevant studies, necessitating more effective literature search methods. Conventional approaches often involve complex search strategies that can be time-consuming and potentially lead to reviewer exhaustion.

The study aimed to assess the effectiveness of four pre-trained sentence transformer models in extracting and comparing semantic similarities between abstracts to assist scholars in selecting articles for systematic reviews. Researchers used a convenience sample of 6110 articles from a previously published systematic review, which had already identified 24 target articles addressing two focused questions (FQ1 and FQ2).

The four models evaluated were all-MiniLM-L6-v2, all-MiniLM-L12-v2, all-mpnet, and All-distilroberta-v1. The models were applied to the dataset to perform semantic similarity searches using the focused questions as queries, calculating cosine similarity to measure how closely related the abstracts were to the FQs.

The findings revealed that the all-mpnet model emerged as the best-performing model, achieving mean similarity scores of 0.77 for FQ1 and 0.80 for FQ2. The all-distilroberta-v1 model also performed well, with scores of 0.76 for FQ1 and 0.78 for FQ2. Even the faster all-MiniLM-L6-v2 model yielded interesting results, identifying all 9 target articles for FQ1 and missing only 2 out of 15 for FQ2 when limiting the search to the top 1000 articles based on similarity scores. The use of these models can significantly reduce the number of articles that need to be manually screened, thus saving time and effort in the systematic review process.

The study suggests that adopting sentence transformers could lead to a shift away from traditional complex search strategies, allowing for broader searches refined by semantic algorithms. Future research directions include exploring the synergy between semantic searches and keyword-based searches to enhance accuracy and reliability, as well as the need for advancements in computing capabilities to enable real-time semantic searches across large databases. In conclusion, the research demonstrates that sentence transformers are a viable approach for streamlining literature searches in systematic reviews, effectively identifying relevant articles, and improving the efficiency of the review process, potentially transforming how researchers approach literature searches in the future [16].

The research paper titled "Multilingual transformer and BERTopic for short text topic modeling: the case of Serbian" by Medvecki, Darija and Bašaragin, Bojana and Ljajić, Adela and Milošević, Nikola (2024) examines the use of BERTopic, an advanced topic modeling method, on brief Serbian texts, particularly tweets expressing COVID-19 vaccine hesitancy. The study utilizes three multilingual embedding models: distiluse, paraphrase, and paraphrase-multilingual-mpnet-base-v2, and assesses their effectiveness under two text preprocessing levels (partial and full). The research explores how preprocessing impacts BERTopic topic representation quality in Serbian,

a morphologically complex language, and compares BERTopic performance to traditional approaches like Latent Dirichlet Allocation (LDA) and Nonnegative Matrix Factorization (NMF) on lemmatized text.

The findings reveal that BERTopic can yield informative topics even with partially preprocessed text, showing a minimal performance drop compared to fully preprocessed text. In terms of Topic Coherence (TC) and Topic Diversity (TD) scores, the best TC score for partially preprocessed text was -0.133 and the best TD score was 0.896 using the paraphrase-multilingual-mpnet-base-v2 model. For fully preprocessed text, the distiluse model achieved the best TC score of -0.042 and shared the best TD score of 0.868 . In comparison, LDA and NMF had TC scores of -0.104 and -0.065 , respectively, and TD scores of 0.897 and 0.795 , respectively.

These results indicate that the BERTopic provides more informative topics and insights, particularly when the number of topics is not limited. The significance of this study lies in demonstrating that BERTopic can be a valuable tool for researchers working with morphologically rich low-resource languages, as it reduces the need for extensive pre-processing and effectively handles short texts. Overall, this article emphasizes the effectiveness of BERTopic in topic modeling for Serbian and its advantages over traditional methods, paving the way for further research in similar linguistic contexts [17].

A comprehensive study conducted by Sarıtaş Karahan, Öz Cahid Arda, and Güngör Tunga offers an in-depth examination of static word embedding models for Turkish, exploring both contextual and non-contextual approaches to assess their efficacy and applications. The research utilized two substantial Turkish corpora, BounWebCorpus and HuaweiCorpus, comprising over 384 million tokens, and implemented various models such as Word to Vector (Word2Vec), FastText, Global Vectors for Word Representation (GloVe), BERT, and Embeddings from Language Models (ELMo). The evaluation framework incorporated both intrinsic and extrinsic methods, including analogy tasks testing semantic and syntactic relationships, whereas extrinsic evaluation employed three downstream NLP tasks: sentiment analysis, part-of-speech tagging, and NER, processing over 341000 training sentences and 41000 test sentences.

The results demonstrated that X2Static BERT consistently outperformed other models across most tasks, while Word2Vec showed particular strength in semantic tasks, and FastText performed well because of Turkish morphological complexity. This study made notable contributions by creating a public repository of Turkish word embeddings, providing evaluation scripts and datasets, and integrating the models into the Turkish Language Processing Platform. This study represents the first comprehensive comparison of static and contextual embeddings for Turkish, establishing important benchmarks for future research in morphologically rich languages. The findings provide practical guidance for model selection in various NLP applications while addressing the unique challenges of processing the agglutinative nature of the Turkish language. This study also identified key areas for future research, including the evaluation of more complex downstream tasks and potential comparisons with other transformer-based models. The comprehensive evaluation framework and detailed results offer valuable insights for researchers and practitioners working with Turkish NLP applications, while the public availability of resources ensures the lasting impact of the study on the field [18].

In the research titled "Selecting Textual Analysis Tools to Classify Sustainability Information in Corporate Reporting," the study presents the efficacy of various NLP methodologies for extracting and classifying sustainability-related information from unstructured text, with a particular focus on 10-K annual reports. The authors compiled a dataset comprising 75000 sentences from the business description and management discussion sections of these reports, which were manually coded to identify sustainability information.

They conducted a comparative analysis of four primary approaches: (1) dictionary-based techniques, which utilize predefined lists of terms; (2) topic modeling, encompassing both supervised and unsupervised methods; (3) word embeddings, such as Word2Vec, that represent words in a continuous vector space; and (4) Large language models (LLMs), including BERT and Chat Generative Pre-trained Transformer (GPT), which generate contextualized embeddings based on the surrounding text. The findings indicated that dictionary methods exhibit significant variability in quality, whereas topic models generally demonstrate superior

performance compared to simpler approaches. Notably, LLM's exhibit the most robust performance, particularly when fine-tuned for specific tasks related to sustainability. This study underscores the critical importance of selecting appropriate NLP techniques to enhance the validity and quality of sustainability information extraction, addressing the increasing demand for reliable corporate sustainability reporting in light of global challenges, such as climate change and biodiversity loss. This research contributes to the understanding of how different textual analysis tools can be effectively utilized in the context of corporate reporting and sustainability [19].

The research paper "Graph-based Nearest Neighbor Search: From Practice to Theory" by Liudmila Prokhorenkova and Aleksandr Shekhovtsov examines the efficacy of graph-based methods in nearest neighbor search. Despite their proven practical success, these approaches have lacked thorough theoretical examination. This study aims to address this gap, particularly in low-dimensional contexts where the dimension d is considerably less than $\log n$ (n being the number of data points), a scenario common in many real-world applications. The study investigates various algorithms, including a fundamental greedy algorithm on nearest neighbor graphs and performance-enhancing heuristics. These heuristics involve adding shortcut edges to expedite searches and maintaining a dynamic list of potential neighbors to enhance accuracy. To evaluate these algorithms, the researchers employ a standard approach, comparing time-versus-quality curves. They measure error rates and distance calculation counts for synthetic datasets while assessing queries per second for real-world datasets.

The paper also explores several experimental techniques, such as k nearest neighbors (k -NN) graph, edge addition based on a specific distribution, prioritizing long links, utilizing beam search, and dimensionality reduction methods. The authors contend that their theoretical insights, backed by experimental outcomes, significantly contribute to understanding the capabilities and limitations of graph-based algorithms. In essence, the article offers a comprehensive analysis of graph-based techniques, blending theoretical rigor with practical observations, which is essential for advancing the field of nearest neighbor search. [20].



2. BACKGROUND

2.1 Large Language Models: An Overview of Language Modeling

Advanced AI systems, known as LLMs, are designed to comprehend, create, and manipulate human languages. These models, built on deep learning architectures, particularly transformer models, can process and generate coherent and contextually appropriate text. By training on extensive text datasets, LLMs can recognize complex patterns, grasp grammar, and develop reasoning capabilities. Their adaptability allows applications in numerous fields, including healthcare, finance, education, and entertainment. LLMs can streamline tasks requiring language comprehension, such as composing emails, creating reports, and responding to inquiries, thereby conserving time and resources. Moreover, their capacity to simultaneously handle substantial amounts of data and interactions makes them suitable for high-demand applications. LLMs also face significant challenges. A primary concern is the potential for bias, as these models can reflect the biases in their training data, potentially resulting in unfair or discriminatory results. This raises important questions about the fairness and accountability of their usage. Moreover, decision-making processes can be difficult to interpret, making understanding how they arrive at specific results challenging, and particularly problematic in critical applications. The substantial computational resources required for training and deploying LLMs present an obstacle to smaller organizations. Lastly, ethical issues arise regarding the possible misuse of LLMs to produce misinformation or harmful content, emphasizing the need for guidelines and regulations to ensure their responsible utilization. In conclusion, while LLMs offer considerable advantages in terms of efficiency and versatility, addressing their limitations is essential to maximize their benefits and minimize potential drawbacks [1].

language model (LM) is an advanced probabilistic system developed to comprehend and produce human language by estimating the probability of word sequences. These models excel because of their structure, with contemporary LMs primarily utilizing the transformer framework that incorporates self-attention mechanisms. This enables the model to assess the relative significance of various words within a sentence, allowing it to grasp long-distance relationships and contextual subtleties in a language. LMs are generally trained on extensive text datasets, learning to forecast subsequent words in a sequence based on preceding words. This training process allows them to absorb the statistical properties of language, including grammar, syntax, factual knowledge about the world, and some degree of reasoning ability. Pre-trained models, such as BERT and GPT series, including GPT-2 and GPT-3, have gained widespread acceptance in the NLP field. These models are frequently fine-tuned for particular tasks such as sentiment analysis, translation, and question answering, making them adaptable tools for various applications.

LMs have shown remarkable capabilities in generating coherent and contextually appropriate text, which improves human-computer interactions in applications such as chatbots and virtual assistants. They can also function as knowledge repositories for cognitive agents, enabling them to extract valuable information and enhance their performance on specific tasks by comprehending the characteristics of the LMs and how to interact effectively with them. Nevertheless, despite their impressive abilities, LMs lack genuine understanding and reasoning capabilities. They produce text based on patterns learned during training and may generate incorrect or nonsensical outputs if the input is ambiguous or outside their training data. This limitation underscores the need for the cautious application and integration of LMs in cognitive systems. LMs represent a powerful advancement in NLP, offering considerable potential for enhancing cognitive agents and various applications when used effectively [21].

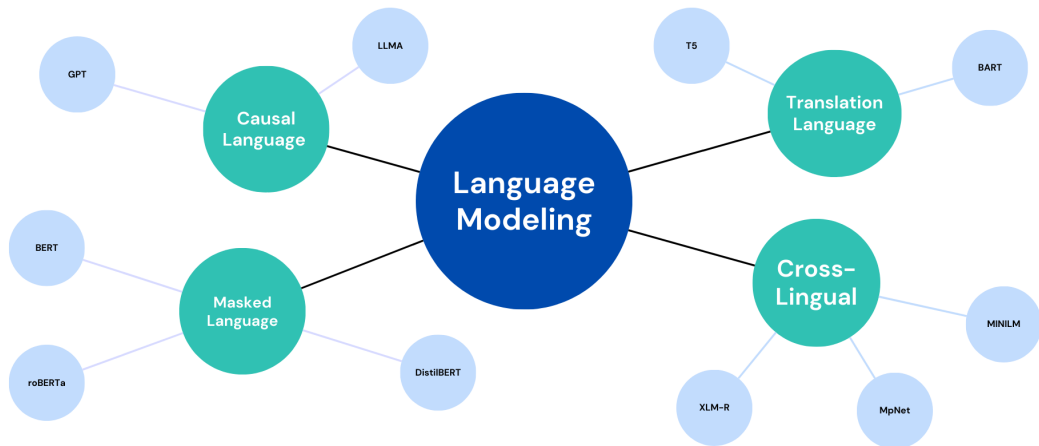


Figure 2.1 : Types of language modeling Representations [1]

2.1.1 Causal language modeling

Causal language modeling (CLM) is an LM that focuses on predicting the next word in a sequence based on the previous word. It operates on the premise that the probability of a word depends only on the words that come before it. This sequential processing allows the model to capture time dependence and context, making it particularly effective for tasks such as text generation. CLM is usually implemented using transformer architectures, which are excellent in dealing with long-distance text dependencies due to their self-aware mechanisms. Examples of models that utilize CLM include the GPT Generative Pre-trained Transformer series, GPT-Neo and GPT-J, OpenAI Codex, Large Language Model Meta AI [4].

2.1.1.1 Generative pretrained transformers

GPT is a model architecture aimed at improving natural language understanding through a two-stage training process. In the first stage, known as generative pretraining, the model is trained on a vast corpus of unlabeled text, which can include diverse sources such as books, articles, and websites. The training objective is to predict the next word in a sentence based on the preceding context, a task that allows the model to learn intricate patterns of language, grammar, and semantics. This

phase is crucial as it enables the model to acquire significant world knowledge and understand long-range dependencies within the text. The transformer architecture, which underpins GPT, employs self-attention mechanisms that allow the model to weigh the importance of different words in a sentence, regardless of their position. Produces embeddings based on the context of the entire input sequence, useful for generating coherent text. This capability is essential for grasping context and meaning, making the model more adept at handling complex linguistic structures compared to traditional recurrent neural networks (RNNs) [22].

After the initial pretraining stage, the model progresses to the second phase: discriminative fine-tuning. During this stage, the pre-trained model is customized for particular tasks using labeled datasets, including question answering, textual entailment, and semantic similarity evaluation. The fine-tuning process incorporates task-specific input modifications, which adapt the model input to better suit each task's requirements. This method facilitates effective transfer learning, allowing the model to utilize knowledge acquired during pretraining while necessitating only minor adjustments to its structure. Fine-tuning typically encompasses hyperparameter optimization, dropout for regularization, and a learning rate strategy to enhance performance on target tasks. The results of using GPT for natural language understanding are impressive. The model has achieved state-of-the-art performance on various benchmarks, demonstrating significant improvements over traditional models [23].

2.1.2 Masked language modeling

MLM is a technique by which a word in a sentence is randomly masked, and the model is trained to predict those masked words based on the context. This approach allows the model to learn the bidirectional context by using information from both sides of the masked token. MLM is a key component of models such as BERT, which have achieved the highest performance in various NLP tasks by effectively understanding the relationship between words in a sentence [4].

2.1.2.1 Bidirectional encoder representations from transformers

BERT is a type of model that helps computers understand the meaning of words in a sentence by looking at the words before and after them. This method allows BERT to better grasp the details of language compared to older models that only read text in one direction. BERT is trained in two main steps: first, during pretraining, it learns from a large amount of text using two tasks. The first task is called the MLM, where some words in a sentence are hidden, and the model learns to guess these words based on the surrounding context. The second task is called NSP, which helps the model learn how two sentences relate by predicting if the second sentence comes after the first in the original text. After this initial training, BERT can be adjusted for specific jobs like answering questions or analyzing feelings, making it adaptable to different tasks.

BERT has shown great improvements in various NLP tasks. For example, it outperformed earlier models on the GLUE benchmark, achieving an average improvement of between 4.5% and 7% over the best-known models at that time. In the SQuAD dataset, BERT reached an F1 score of 83.1, surpassing previous models and getting close to human-level performance. There are also special versions of BERT, like Biomedical BERT and Scientific BERT, created for specific fields such as medicine and science, showing better results on tasks related to those areas. A metric called BERTScore was introduced to measure how good generated text is, by calculating the similarity between words in reference and candidate sentences, and it aligns better with human opinions than older measurement methods.

Although BERT greatly boosts performance, it requires a lot of computing power; training the full version can take days on strong machines, which is something to think about for real-world use. Overall, BERT marks a major step forward in NLP, creating a strong tool for understanding and producing human language, with uses in many fields and tasks. Its ability to be specifically fine-tuned makes it even more useful and effective [24].

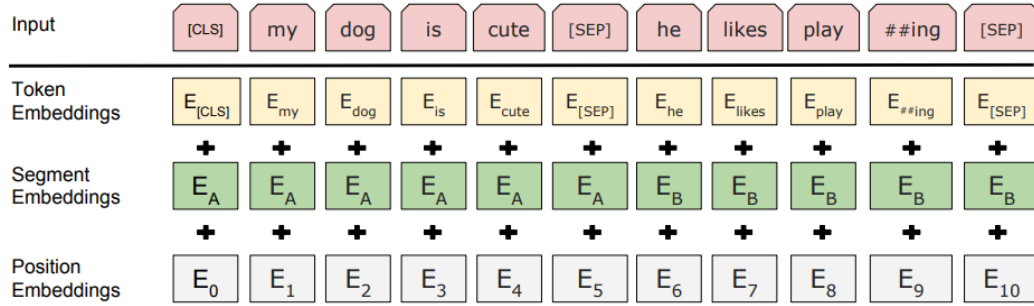


Figure 2.2 : BERT Input Representation [2]

In Figure 2.2, the BERT model's input representation architecture is illustrated, which employs a sophisticated three-layer architecture for sentence processing, where the raw textual input occupies the uppermost layer, followed by three critical embedding layers in the intermediate section. These embedding layers consist of token embeddings for semantic representation, segment embeddings for structural attribution, and positional embeddings for spatial context, which are collectively aggregated to generate a comprehensive input representation. This integrated approach enables BERT to simultaneously evaluate semantic meaning, structural position, and segment attribution, facilitating exceptional performance in NLP tasks through its ability to capture both local and global dependencies within the text.

2.1.2.2 Robustly optimized BERT pretraining approach

Robustly Optimized BERT Pretraining Approach (RoBERTa), developed by Facebook AI researchers, is an advanced LM that builds upon and enhances BERT's foundation. What makes RoBERTa stand out is its significantly longer training period compared to BERT, allowing the model to develop deeper language understanding. This extended training is augmented by larger batch sizes, enabling the model to learn from more diverse examples in each training iteration.

A key modification in RoBERTa is the removal of BERT's NSP task. The researchers found this task didn't meaningfully contribute to model performance, so they simplified training to focus solely on MLM. RoBERTa introduces dynamic masking, where tokens are randomly masked during each training epoch, rather than using static

masking patterns. This approach exposes the model to various contexts for the same tokens, enhancing its language comprehension abilities.

RoBERTa processes longer text sequences, improving its ability to maintain context and understand relationships across extended passages. The model leverages extensive training data, including the CC-NEWS dataset, exposing it to diverse linguistic patterns. While retaining BERT's transformer architecture, RoBERTa offers two configurations: BASE (12 layers, 768 hidden sizes, 12 attention heads) and LARGE (24 layers, 1024 hidden size, 16 attention heads).

The model has demonstrated exceptional performance on various benchmarks including GLUE, RACE, and SQuAD, surpassing both BERT and contemporary models. This success validates RoBERTa's design choices and establishes it as a significant advancement in NLP, showing how methodical improvements in training approaches and architecture can substantially enhance model capabilities [25].

2.1.2.3 Distillation bidirectional encoder representations from transformers

Distillation BERT (DistilBERT) is a groundbreaking model designed as a condensed version of BERT, which has revolutionized NLP. While maintaining impressive language comprehension capabilities, DistilBERT achieves remarkable efficiency improvements, featuring a 40% reduction in size and a 60% increase in processing speed compared to its predecessor. This optimization makes it particularly valuable for resource-limited environments. The model's most notable achievement is retaining 97% of BERT's language understanding capabilities despite its smaller footprint. This is accomplished through knowledge distillation, an innovative training process where DistilBERT learns to emulate BERT's behavior. During training, DistilBERT actively learns from BERT's outputs, enabling it to excel in various NLP tasks including sentiment analysis, question answering, and NER, sometimes even outperforming the original BERT model.

The training methodology employs a sophisticated triple-loss function system. First, the LM loss focuses on the next-word prediction in sequences. Second, the distillation loss ensures DistilBERT's output probabilities closely match BERT's predictions.

Third, the cosine-distance loss works to maintain similar embedding patterns between the two models, preserving the semantic integrity of learned representations. Built on the Transformer architecture, DistilBERT leverages self-attention mechanisms to effectively process and understand contextual relationships in text. This architectural choice allows the model to be fine-tuned for specific applications while maintaining its efficiency advantages. The model's reduced computational requirements make it especially suitable for mobile devices and real-time processing systems, where resource optimization is crucial.

DistilBERT's practical applications extend across various domains, particularly benefiting scenarios requiring quick processing or operating under hardware constraints. Its ability to deliver high-quality NLP capabilities with reduced computational overhead makes it an attractive choice for developers implementing advanced language processing features in resource-conscious environments. This combination of efficiency and performance represents a significant advancement in making sophisticated LMs more accessible and practical for widespread deployment [26].

2.1.3 Translation language modeling

Translation Language Modeling (TLM) is an extension of MLM that uses parallel sentences in multiple languages to train. In TLM, the model is trained on parallel sentence pairs, randomly removing words in source and target sentences. This allows the model to address both the source and target contexts in the prediction of masked words, thereby facilitating the alignment of representations across languages. TLM is particularly useful in tasks such as machine translation, where understanding the relationships between different languages is crucial [4].

2.1.3.1 Text to text transfer transformer

The T5 represents a revolutionary development in NLP that transforms all linguistic tasks into text-to-text conversion problems. The model's distinctive feature lies in its universal approach: every input and output is formatted as text, regardless of whether the task involves translation, summarization, or question answering. The

model operates through a dual-stage process: pretraining followed by fine-tuning. The initial pretraining phase exposes the model to a comprehensive text dataset, utilizing a denoising autoencoder technique where the model learns to reconstruct partially masked text inputs. This foundation is then refined through task-specific fine-tuning, where the model adapts its pre-trained knowledge to particular applications using labeled datasets.

A fundamental advantage of T5 lies in its transfer learning capabilities. The broad exposure during pretraining enables the model to develop versatile features that can be adapted to specific tasks through fine-tuning, allowing excellent performance even with limited task-specific training data. The model's architecture comes in various sizes, offering flexibility to match different computational resources and performance needs.

T5's significance stems from its success in unifying diverse NLP tasks within a single framework. The model achieves exceptional results across various natural language understanding and generation tasks through its innovative text-to-text methodology, comprehensive training approach, and effective transfer learning capabilities. This unified approach has established new benchmarks in the field, demonstrating the effectiveness of addressing complex language tasks through a standardized framework [3].

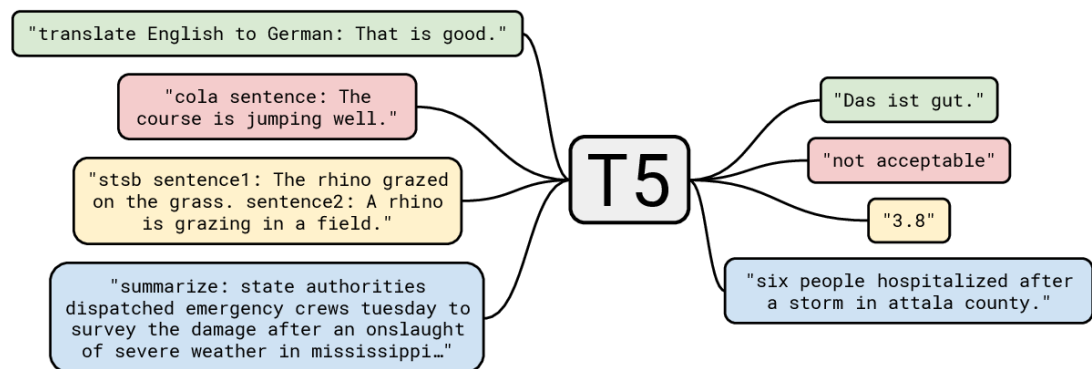


Figure 2.3 : Illustration of Text-to-Text Framework [3]

Figure 2.3 illustrates a text-to-text system architecture where any given language task including translation, question answering, and classification is approached through a unified framework: the model receives text input and produces text output. This standardized approach enables the use of consistent model architecture, loss function, and hyper-parameters across different types of language tasks. This framework also serves as a uniform testing environment for evaluating various methods in their empirical research. The central component labeled "T5" stands for "Text-to-Text Transfer Transformer," which is their proposed model that implements this unified approach.

2.1.4 Cross-lingual language model

Cross-lingual Language Model (XLM) combines different training objectives to create a robust system for processing multiple languages. At its core, XLM uses both monolingual and parallel data in clever ways: it can train on single-language texts using CLM and MLM methods, while also learning from translated texts through TLM. By switching between these approaches, XLM develops two crucial abilities it learns what makes each language unique while also figuring out how to match up meanings across different languages. This makes it especially good at tasks like translation and cross-language classification, where it has set new performance standards.

The big picture is that XLM represents a major step forward in helping computers work with multiple languages at once. By being able to handle both language-specific details and cross-language connections, it's become an important tool for tackling complex language processing challenges across linguistic boundaries.

In essence, LMs have become essential building blocks in modern NLP, giving machines the ability to work with human language in sophisticated ways. The field's progress from basic statistical approaches to advanced systems like XLM shows how far we've come in developing technology that can help break down language barriers and improve communication between different linguistic communities [4].

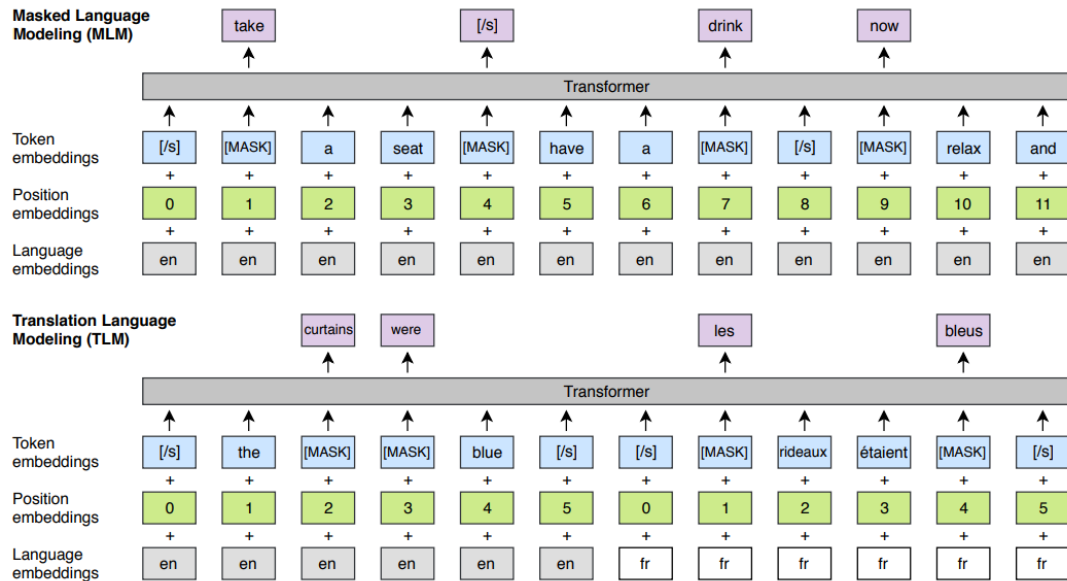


Figure 2.4 : XLM Pretraining Architecture [4]

The 2.4 illustration outlines a dual-system approach to multilingual language processing and understanding. The first system employs MLM, where individual language text processing occurs through strategic word masking and position tracking within a Transformer structure. This foundation is enhanced by TLM, which introduces parallel sentence processing across languages, maintaining separate language identities while allowing cross-language information flow. The brilliance of this design lies in its simultaneous processing capability - when working with masked words, the system can leverage context from multiple languages at once. This innovative combination has led to remarkable advancements in cross-language processing and translation capabilities, setting new standards in multilingual NLP.

2.1.4.1 Cross-lingual language model-roberta

The Cross-lingual Language Model-Roberta (XLM-R) series encompasses three XLMs of increasing size and capability. Starting with XLM-R Large, containing 550 million parameters, this initial version established the groundwork for cross-lingual processing. Though effective in many applications, its performance is surpassed by its successors [27].

The mid-tier variant, XLM-R XL, expanded to 3.5 billion parameters, enabling more sophisticated language processing. This larger scale yielded tangible benefits,

notably a 1.8% accuracy boost on Cross-lingual Natural Language Inference (XNLI) evaluations compared to its predecessor, setting new performance records across multiple languages.

At the pinnacle of the series stands XLM-R XXL, with a massive 10.7 billion parameter. This version represents a major advancement, showing improved results for both widely-spoken and less common languages. It exceeds XLM-R XL's XNLI performance by 2.4% and even outperforms RoBERTa-Large on English-specific tasks. This model demonstrates how increasing model size can lead to better multilingual capabilities, making it particularly valuable for language processing applications.

2.1.4.2 Masked and permuted pretraining for language understanding

Masked and Permuted Pretraining (MPNET) for Language Understanding, is a cutting-edge pretraining technique that aims to overcome the shortcomings of earlier models such as BERT and Generalized Autoregressive Pretraining for Language Understanding (XLNet) by integrating their advantages. The main objective of MPNet is to enhance language comprehension by tackling two crucial issues: the interdependence of predicted tokens and the positional inconsistencies between the pretraining and fine-tuning stages.

MPNet employs MLM, akin to BERT, where specific tokens in a sentence are concealed, and the model is trained to forecast these hidden tokens based on the surrounding context. This method enables the model to acquire bidirectional language representations, utilizing context from both sides of the masked tokens. Furthermore, MPNet incorporates Permuted Language Modeling (PLM), inspired by XLNet, which allows the model to predict tokens in a shuffled order instead of a fixed sequence. This approach helps capture token dependencies more efficiently, as permuting the input sequence enables MPNet to learn from diverse contexts, thereby improving its grasp of word relationships.

A notable innovation in MPNet is its utilization of auxiliary position information, which provides the model with supplementary positional encodings to aid in

understanding token order within a sentence. This feature addresses the positional discrepancy problem faced by XLNet, where training and fine-tuning phases may not align in terms of token positions. MPNet undergoes pretraining on an extensive dataset comprising over 160GB of text data, allowing it to acquire rich language representations. Following pretraining, the model is fine-tuned on various downstream tasks, including the GLUE benchmark and the SQuAD, which helps it adapt to specific tasks and enhances its performance.

Regarding performance metrics, MPNet has shown superior results on several NLP benchmarks compared to its predecessors. For example, on the GLUE benchmark, MPNet surpassed BERT by 4.8 points, XLNet by 3.4 points, and RoBERTa by 1.5 points. In reading comprehension tasks like SQuAD, MPNet achieved higher accuracy rates than previous models, demonstrating its effectiveness in understanding context and answering questions.

MPNet can be applied to a wide range of NLP tasks, such as text classification, sentiment analysis, NER, question answering, and text summarization. In summary, MPNet represents a significant advancement in the field of NLP. By effectively combining MLM and PLM techniques and addressing the limitations of previous models, MPNet enhances language understanding and achieves state-of-the-art performance on various benchmarks, making it a powerful tool for numerous NLP applications [28].

2.1.4.3 Mini language model

Mini Language Model (MINILM) is a creative framework designed to compress pre-trained Transformer models without sacrificing their performance across various NLP tasks. This task-agnostic approach aims to create smaller, more efficient models that maintain the capabilities of their larger counterparts in areas such as question-answering and summarization.

The framework employs a teacher-student model, where a larger pre-trained model guides a smaller one. The student model learns by mimicking the teacher's

self-attention distributions and hidden states, enabling it to capture crucial contextual information essential for effective language understanding.

A notable feature of MINILM is its use of soft-label distillation. This method allows the student model to learn from the teacher’s probability distributions, providing more detailed insights into decision-making and enhancing its ability to generalize across tasks. MINILM utilizes a uniform layer mapping strategy between teacher and student models, simplifying layer alignment and facilitating effective knowledge transfer.

MINILM’s architecture is adaptable, offering configurations such as 12-layer and 6-layer models with a hidden size of 384. The training process involves optimizing the student model to minimize the difference between its outputs and those of the teacher, typically using a loss function that measures the Kullback-Leibler divergence between their attention distributions.

Extensive evaluations of MINILM on various NLP tasks, including SQuAD 2.0 and GLUE benchmarks, as well as natural language generation tasks like question generation and abstractive summarization, have shown promising results. The framework has demonstrated competitive performance compared to larger models like BERT and RoBERTa, while offering significant improvements in computational efficiency. For example, the 12-layer MINILM model has shown a 2.7 speedup in training while outperforming BERT on certain benchmarks.

MINILM’s ability to create compact yet powerful models makes it particularly suitable for deployment in resource-constrained environments where smaller model sizes and faster inference times are crucial. By effectively distilling knowledge from larger pre-trained Transformers into smaller, efficient models, MINILM offers a balanced approach to NLP applications, combining effectiveness with computational efficiency [29].

2.2 Text Embeddings

Text embeddings are numerical vector representations that capture the semantic essence of the text, facilitating various NLP applications such as retrieving information, answering questions, and determining semantic similarity between texts [30]. These embeddings encode the significance of words, phrases, or complete sentences in a machine-comprehensible format, enabling efficient text data processing and comparison. Early techniques for creating text embeddings include Latent Semantic Indexing, which employs singular value decomposition to reduce the term-document matrix dimensions, as well as Word2Vec, which learns word representations based on their surrounding context in extensive corpora, and GloVe, which captures global word statistics within a corpus [31].

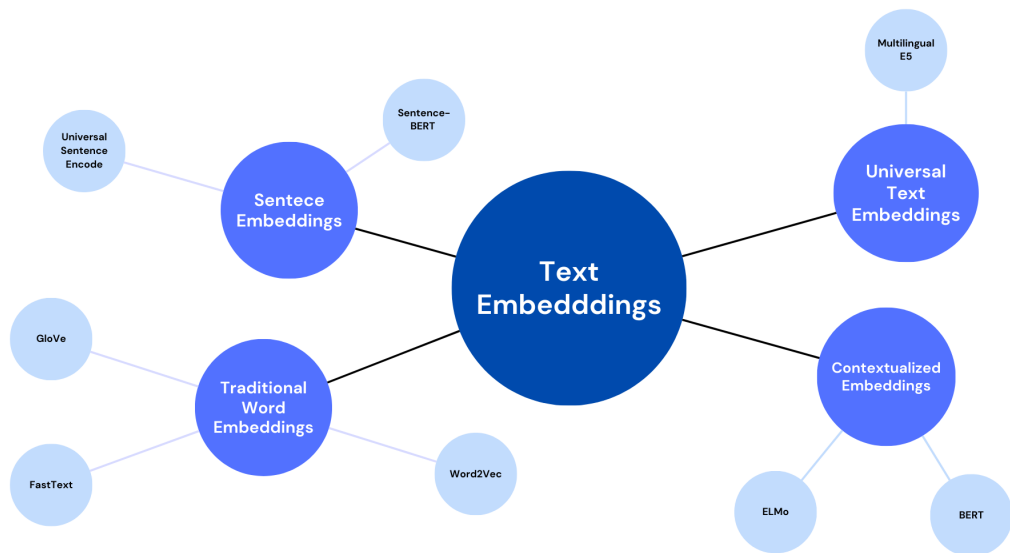


Figure 2.5 : Type Of Text Embeddings Models

The advent of pre-trained LMs like BERT has markedly enhanced text embedding quality. Models such as Sentence BERT and Simple Contrastive Sentence Embeddings adapt BERT for specific tasks to generate more contextually aware embeddings [32]. Contemporary state-of-the-art methods, including Embedding for Everything (E5) and BGE, employ a multi-phase training strategy, initially pretraining on large datasets of weakly-supervised text pairs, followed by fine-tuning on smaller, high-quality labeled datasets to boost performance [33]. However, these multi-phase approaches

often necessitate intricate training pipelines and depend on manually curated datasets, potentially restricting task diversity and language coverage, particularly in models that predominantly focus on high-resource languages.

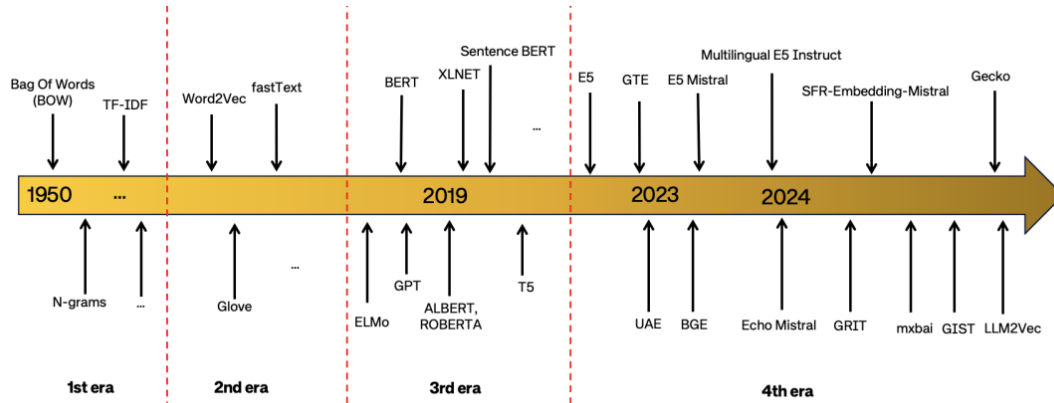


Figure 2.6 : Different Eras of Text Embeddings [5]

This 2.6 shows the evolution of text embedding methods from 1950 to 2024, divided into four distinct eras. The first era (1950s-early 2000s) introduced basic count-based methods like Bag of Words and Term Frequency-Inverse Document Frequency. The second era brought traditional word embeddings with models like Word2Vec and GloVe. The third era around 2019 marked a significant shift toward contextualized embeddings with transformer-based models like BERT and XLNET. Finally, the fourth era (2023-2024) represents the current state of universal text embeddings, featuring advanced models like E5, General Text Embeddings, and SFR-Embedding-Mistral. The timeline effectively illustrates the progression from simple statistical approaches to sophisticated universal embedding models that can handle various tasks and domains.

2.3 Word Embeddings

Word embeddings, also called word Vectors, are essential numerical representations of words in a continuous vector space. These representations allow computers to process and interpret human language more effectively by positioning words as points in a high-dimensional space, where the proximity between points indicates semantic similarity. This approach captures nuanced relationships and meanings that

are challenging to convey using conventional methods like one-hot encoding, which assigns a unique integer to each word.

Word representations come in two main forms: word type vectors and word token vectors. Word type vectors provide fixed representations for each word type, meaning every instance of a word has the same vector regardless of its context. This method can result in ambiguity for words with multiple meanings, such as "bank," which could refer to a financial institution or a riverbank. Conversely, word token vectors are context-sensitive representations that allow each word instance (token) to have a distinct vector based on its surrounding context, offering a more refined understanding of meaning.

The development of word embeddings has advanced from basic frequency-based techniques to more sophisticated neural network approaches. Initial methods relying on co-occurrence statistics paved the way for advanced techniques like word2vec and ELMo, which utilize deep learning to generate contextual embeddings. Recent progress has led to the creation of contextual word embeddings that dynamically adjust a word's representation based on its sentence context. For example, the word "bark" would have different vectors when used about a tree versus a dog. This method addresses the limitations of static word type vectors and provides a more precise representation of meaning [34].

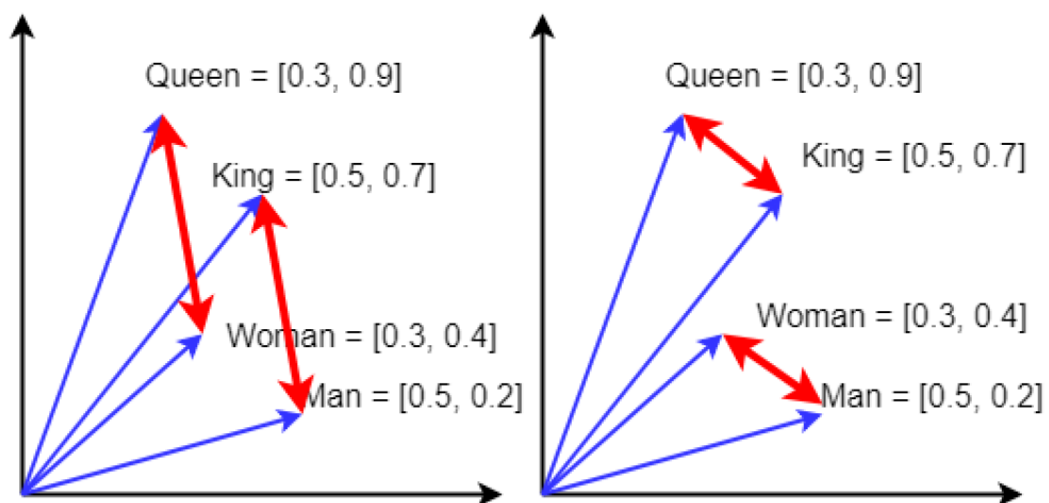


Figure 2.7 : The King-Queen Relationship in 2D Embedding Space [6]

Word embeddings' ability to capture semantic relationships is demonstrated in the 2.7 figure. The illustration presents two 2D coordinate systems that plot four key vectors: King $[0.5, 0.7]$, Queen $[0.3, 0.9]$, Man $[0.5, 0.2]$, and Woman $[0.3, 0.4]$. The diagram reveals how mathematical operations on word vectors can uncover meaningful linguistic patterns through red arrows highlighting analogical connections and blue arrows showing base vectors. Although simplified for 2D visualization, this representation effectively shows how embeddings preserve semantic relationships through the relative positioning of words in vector space, with parallel arrows indicating similar relational meanings between pairs of words.

Word embeddings are utilized in various NLP tasks, including translation, sentiment analysis, and information retrieval. They enable models to perform mathematical operations on words, revealing relationships. In conclusion, word embeddings are a fundamental NLP concept that facilitates language representation in a manner that captures semantic relationships and contextual meanings, significantly enhancing machines' ability to process and comprehend human language.

2.3.1 Traditional word embeddings

Traditional Word embeddings revolutionized NLP by converting words into compact numerical vectors that better capture word meanings and relationships compared to basic methods like one-hot encoding. The main embedding techniques Word2Vec, GloVe, and FastText create dense vectors in a lower-dimensional space, making calculations more efficient and improving how well the models work on new data [35].

Word2Vec works in two ways: Continuous Bag of Words (CBOW) tries to guess a word from the words around it, while Skip-Gram does the opposite - it predicts surrounding words from a target word. Both approaches use neural networks trained on large text datasets, helping capture their meanings and similarities [31].

GloVe takes a different route by analyzing how often words appear together in text and creating embeddings that preserve these patterns. Generates embeddings by capturing global statistical information of words in a corpus [36].

FastText builds on Word2Vec by looking at parts of words, which helps it handle new words and recognize similarities in word structure. The Facebook AI Research team created FastText, a high-performance text analysis tool that efficiently handles both text classification and word representation. What sets it apart is its unique approach of breaking words into smaller character sequences, combined with skip-gram modeling for understanding word relationships. By implementing advanced techniques like hierarchical softmax and negative sampling, FastText achieves remarkable processing speeds on basic Central Processing Unit (CPU) hardware, handling billions of words in minutes. This library matches the accuracy of sophisticated deep learning systems like Convolutional Neural Networks and RNNs but processes data significantly faster, making it particularly valuable for tasks ranging from sentiment analysis to document classification [37].

These methods are based on the idea that words with similar meanings appear in similar contexts, so they end up close to each other in the vector space. This has made them valuable for many language tasks like analyzing sentiment and translating between languages. However, these traditional embeddings struggle with understanding long-range relationships between words and dealing with words that have multiple meanings. Despite these limitations, word embeddings marked a crucial step forward in NLP, laying the groundwork for many applications and continuing research in the field [35].

2.3.2 Contextualized embeddings

Contextual embeddings are sophisticated word representations that capture word meanings based on their context within text. Unlike traditional embeddings that assign fixed vectors to words, these generate context-dependent representations for more nuanced language understanding. Starting with ELMo's bidirectional long short term memories and advancing to BERT's transformer architecture, these models can process entire sequences simultaneously to understand complex word relationships.

The technology relies on deep learning architectures, especially transformers with attention mechanisms that weigh word importance in sentences. These models undergo pretraining on large datasets using unsupervised learning and can be fine-tuned for

specific tasks while maintaining their pre-trained knowledge. Key benefits include handling multiple word meanings, better NLP benchmark performance, and enabling knowledge transfer between tasks. While challenges exist around computational requirements, interpretability, and potential bias, ongoing research works to address these through model compression and multilingual embedding development. This technology represents a major step forward in helping computers better understand and generate human language [38].

2.3.3 Universal text embeddings

Universal Text Embeddings are about creating a single method for generating text representations that work well for many different language tasks. The aim is to develop embeddings that can be used in various situations, like checking how similar texts are, grouping them, searching, and reordering results, without needing changes for each specific task. However, it is difficult to achieve this because different methods often work better for certain tasks but not for others. This variation shows how complex text representation is and emphasizes the need to better understand how embeddings function across different tasks. Researchers are continuously working on universal text embeddings, testing new structures and training techniques to make these embeddings more useful for many applications. The ultimate goal is to create a universal text embedding that performs reliably across various tasks, making it easier to use models in real-life situations without having to retrain or fine-tune them for different purposes [39].

2.3.3.1 Multilingual E5 text embeddings

Multilingual E5 text embeddings (mE5) are advanced models created to improve information retrieval systems and retrieval-augmented LMs across various languages. These models, known as mE5-small/base/large, tackle the shortcomings of existing embeddings that mainly concentrate on English, allowing for improved performance in multilingual settings.

The development of these embeddings involves a two-phase approach. The initial stage consists of weakly-supervised contrastive pretraining on a diverse set of roughly

1 billion multilingual text pairs from various sources, including Wikipedia and mC4. Subsequently, the models undergo fine-tuning on a smaller collection of high-quality labeled datasets. An additional variant, mE5-large-instruct, incorporates synthetic data produced by models such as GPT-3.5/4, further enhancing its capabilities.

The models come in three sizes (small, base, and large), allowing users to choose based on their needs for inference efficiency and embedding quality. The instruction-tuned model is particularly noteworthy for its performance, which is on par with cutting-edge English-only models.

mE5 models can be utilized for various tasks, such as information retrieval, semantic similarity, and clustering across numerous languages, making them versatile tools for practitioners. The model weights are publicly accessible, enabling users to employ these embeddings in their applications. In summary, the mE5 represents a major step forward in the domain of multilingual text processing [40].

2.4 Sentence Embeddings

Sentence embeddings are compact vector forms that represent the meaning of sentences, making them useful for various tasks in NLP, such as classifying text and measuring semantic similarity. They work well in transfer learning, enabling models to achieve good results even when there's not much-supervised training data available, which helps when large labeled datasets are scarce. Often, sentence embeddings perform better than traditional word-based transfer methods and can be used in many NLP applications, such as scoring semantic similarity and retrieving information. Overall, they mark an important step forward in making language data processing more efficient [41].

2.4.1 Sentence BERT

Sentence BERT (SBERT), is designed to create meaningful sentence embeddings that can be applied to various NLP applications. These embeddings allow sentences to be compared in vector space, making them valuable for tasks like semantic searching, clustering, and retrieving information. SBERT was developed to produce

higher-quality embeddings than conventional methods, making it easier to identify sentences with similar meanings.

SBERT functions by modifying BERT's architecture to incorporate either a Siamese or triplet network configuration. In the Siamese setup, identical networks process sentence pairs, while triplet networks handle three sentences (anchor, positive, and negative) to understand their relationships. The model undergoes fine-tuning using specialized semantic textual similarity datasets, adapting the pre-trained BERT model to better understand sentence relationships.

The system generates consistent-sized embeddings that can be evaluated using various similarity metrics, including cosine similarity and distance measurements like Manhattan or Euclidean. This approach significantly improves computational performance for example, clustering 10,000 sentences takes about 5 seconds with SBERT compared to 65 hours using traditional BERT.

SBERT and similar Sentence Transformers have numerous practical applications. They excel in semantic search systems, enabling the identification of meaningfully similar sentences - crucial for search engines, recommendation systems, and chatbots. The embeddings can group sentences or documents by meaning, helping organize large datasets and summarize information. They're also valuable for classification tasks like sentiment analysis and topic categorization.

These transformers enhance information retrieval by focusing on semantic understanding rather than simple keyword matching. They've shown exceptional results in specialized tasks like argument similarity analysis, where they can effectively differentiate between various aspects of textual arguments. Overall, Sentence Transformers like SBERT represent a powerful advancement in creating high-quality sentence embeddings, substantially improving both effectiveness and efficiency in semantic processing tasks [42].

2.4.2 Universal sentence encoder

The Universal Sentence Encoder (USE) is an advanced model created by Google researchers to convert sentences into fixed-size embedding vectors. These vectors help improve transfer learning in many NLP tasks. The main purpose of the USE is to deliver a strong representation of sentences that can be used in various applications without needing a lot of retraining. This is especially helpful when there isn't much-labeled data available, allowing users to use pre-trained models for good results on new tasks.

The USE has two main types: the transformer-based encoder and the Deep Averaging Network (DAN). The transformer model uses self-attention techniques that help it determine how important certain words are in a sentence based on their context. This leads to representations that reflect the deeper meanings and relationships among words, resulting in high accuracy for various NLP tasks. However, it needs more computing power, making it more demanding on resources.

In contrast, the DAN provides a simpler option. It creates sentence embeddings by averaging the word embeddings of the words in a sentence. This method is quicker and more efficient, although it might not be as accurate as the transformer model. The choice between these models often depends on whether speed or accuracy is more important for a specific application.

The output of the Universal Sentence Encoder is a fixed-size vector that captures the meaning of the input sentence. These embeddings are flexible and can be used for many tasks, such as measuring how similar two sentences are in meaning, classifying text into specific categories, grouping similar sentences, and finding relevant documents based on user searches.

The USE has been thoroughly tested on various benchmarks, including the STS Benchmark, where it has performed well. This shows its ability to understand the complexities of natural language, making it a useful tool for researchers and professionals. By turning sentences into meaningful vector representations, the Universal Sentence Encoder supports a variety of applications in NLP, such as

improving search engines and making chatbots and virtual assistants better. Its ability to adapt to different tasks is a key part of modern NLP solutions [41].

2.5 Text Preprocessing

Text pre-processing is a crucial step in NLP that aims to transform raw data into a structured format by eliminating words that don't contribute to the sentence's contextual meaning. Various techniques are available for this process, and the selection of methods depends on the specific problem at hand. However, certain techniques are essential for improving model performance across all problems. These techniques should be applied in a particular sequence to achieve optimal results.

Segmentation involves dividing a continuous text block into meaningful sentences or segments, typically using punctuation marks as delimiters. This step may not always be necessary and depends on the text corpus size. The removal of punctuation marks, special characters, and URLs is essential for cleaning raw text data, as these elements often don't add to the text's meaning. Converting all text to lowercase helps standardize the content, ensuring uniform treatment of words regardless of their original case. This process aids in reducing data redundancy. Tokenization breaks down text into individual words or tokens, which is vital for subsequent analysis and feature extraction, allowing models to work with discrete text units. Parts of speech tagging assigns grammatical categories to each token, helping to understand the text's grammatical structure, which is beneficial for various NLP tasks. Eliminating stopwords, which are common words with little semantic value, helps focus on more meaningful words in the analysis, reducing noise in the data. Text normalization standardizes text to remove noise, particularly in informal contexts like social media. It involves converting variations of words to their standard form. Stemming reduces words to their root form, which may not always be a valid word. While useful in some applications like search engines, it can sometimes result in a loss of semantic meaning. Lemmatization, similar to stemming, reduces words to their base form (lemma) while preserving meaning. It is often preferred over stemming due to its ability to maintain semantic integrity [43].

Additionally, noise removal is an essential aspect of text cleaning, which involves eliminating irrelevant characters, numbers, or symbols that do not contribute to the analysis. This cleaning process enhanced the relevance of the dataset by ensuring that only meaningful information was retained. However, a significant disadvantage of noise removal is that important information may be inadvertently removed if the criteria for what constitutes "noise" are not carefully defined. For instance, in a dataset containing product reviews, removing all numbers can eliminate valuable information regarding ratings or prices [44].

2.6 Similarity Search

Similarity search is a computational technique designed to identify items within a dataset that most closely resemble a given query item. The main objective is to locate the k-NN of a query vector among a set of high-dimensional vectors, typically employing a distance measure such as L2 distance. This method is extensively utilized across various fields, particularly in specialized database systems handling complex data types like images and videos, making it crucial for applications such as image retrieval, recommendation systems, and content-based search.

Nevertheless, similarity search encounters substantial obstacles, especially when dealing with extensive datasets containing billions of entries. Major challenges include the curse of dimensionality, which hinders exhaustive search and precise indexing, as well as the considerable memory requirements of existing methods that can restrict scalability. To tackle these issues, researchers have developed optimized algorithms that harness GPU architecture to enhance similarity search performance. These approaches improve parallelism and optimize memory hierarchy usage, resulting in substantial speed enhancements compared to previous cutting-edge techniques.

In summary, similarity search remains a vital area of research and application, particularly in efficiently managing high-dimensional data and facilitating rapid retrieval of similar items from large collections [45].

2.6.1 Linear search

Linear search, also known as sequential search, is a basic algorithm used to find a particular value in an array or list by examining each element sequentially. The process begins at the start of the list, where the algorithm checks each item in sequence until it either finds the target value or reaches the end of the list. With a time complexity of $O(n)$, linear search may need to examine every element in the worst case, rendering it less efficient for extensive datasets compared to more sophisticated search techniques such as binary search. Nevertheless, its uncomplicated nature and the fact that it doesn't require sorted data make linear search a suitable option for small or unsorted datasets. This straightforward method is easily implementable across various programming languages, making it a popular introductory example in computer science courses [46].

2.6.2 Radial basis function

Radial Basis Function (RBF) neural networks are a distinct class of artificial neural networks featuring a three-layer structure: an input layer, a hidden layer employing nonlinear radial basis functions (typically Gaussian), and a linear output layer. These networks demonstrate proficiency in various applications, including nonlinear system identification, pattern recognition, and function approximation, by effectively capturing intricate data relationships. The training process involves determining the RBF centers and widths, as well as optimizing output layer weights, often utilizing methods such as k-means clustering and orthogonal least squares. The article highlights that the introduced innovative adaptive kernel improves RBF network performance by dynamically adjusting kernel weights, yielding superior outcomes compared to conventional kernel fusion techniques [47].

2.6.3 Manhattan distance

The distance between two points measured along paths that follow a grid pattern is known as Manhattan distance. This metric, which also goes by the names "taxicab

distance" or "city block distance," adds up the absolute differences between the points' coordinate values [48]. The Formula of Manhattan Distance is

$$d(x,y) = \sum_{i=1}^n |x_i - y_i| \quad (2.1)$$

where

- $d(x,y)$ represents the Manhattan distance between two points x and y
- $|x_i - y_i|$ represents the absolute difference between coordinates
- $i = 1$ to n indicates we're summing across all dimensions (n dimensions) [49]

2.6.4 Cosine similarity

Cosine similarity serves as a measurement tool for evaluating the similarity between two vectors within a multi-dimensional space, often utilized in textual analysis. This method computes the cosine of the angle formed by the vectors, producing a result ranging from -1 to 1 . A cosine similarity of 1 indicates that the vectors are identical, while a value of 0 signifies that they are orthogonal, meaning there is no similarity. A value of -1 denotes that the vectors are completely dissimilar. This technique proves especially useful in comparing documents, as it emphasizes the direction of the vectors rather than their size, making it particularly effective for examining texts of different lengths. By portraying documents as vectors in a high-dimensional space, cosine similarity offers a dependable approach to quantify the resemblance of textual content [50]. The mathematical formula for cosine similarity is

$$\text{CosineSimilarity} = \frac{D_1 \cdot D_2}{\|D_1\| \|D_2\|} \quad (2.2)$$

where

- $D_1 \cdot D_2$ is the dot product of the vectors $D1$ and $D2$.
- $\|D_1\|$ is the magnitude of vector $D1$.
- $\|D_2\|$ is the magnitude of vector $D2$.

The dot product is calculated as:

$$D_1 \cdot D_2 = \sum_{i=1}^n D_{1i} \times D_{2i} \quad (2.3)$$

And the magnitude of a vector D is calculated as:

$$\|D\| = \sqrt{\sum_{i=1}^n D_i^2} \quad (2.4)$$

2.6.5 Euclidean distance

The Euclidean distance quantifies the direct path between two points in Euclidean space. This measurement is extensively used across various disciplines, including geometry, physics, and machine learning, due to its straightforward interpretation as the shortest distance between two points. The Euclidean metric adheres to the fundamental properties of a metric, such as non-negativity, identity of indiscernibles, symmetry, and triangle inequality. Euclidean distance is a commonly utilized metric that determines the linear distance between two points in Euclidean space. In contrast to other distance measures like Manhattan distance, which calculates the sum of absolute differences, Euclidean distance emphasizes the direct line between points, thereby influencing clustering outcomes based on data distribution [51]. The formula is

$$D(p1, p2) = \sqrt{\sum_{i=1}^N (p1[i] - p2[i])^2}. \quad (2.5)$$

where

- $p1$ and $p2$ represent two probability distributions
- $D(p1, p2)$ represents the distance between $p1$ and $p2$

2.7 Vector Indexing

The organization and retrieval of high-dimensional data represented as vectors, known as vector indexing, is a vital technique in fields such as machine learning, image processing, and geographic information systems. Its importance stems from its capacity to boost efficiency and scalability when dealing with large data sets. As data volumes continue to expand, effective indexing methods are crucial for maintaining performance, allowing systems to handle complex queries in real time. Vector indexing substantially improves search performance by structuring data in a way that minimizes the search space, which is essential for applications like recommendation systems and image retrieval.

Although vector indexing offers benefits such as faster query response times and adaptability to multidimensional data management, it also presents challenges. Implementing and maintaining these indexing structures is complex and requires careful consideration of data characteristics and query patterns. Moreover, the "curse of dimensionality" can negatively impact the performance of certain indexing methods as the number of dimensions grows, leading to inefficiencies. In addition, indexing structures may require additional storage space, which can be problematic in environments with limited resources.

In summary, vector indexing plays a critical role in the management of high-dimensional data, providing significant advantages in terms of efficiency and performance. However, it also presents challenges that must be addressed in practical applications [52].

2.7.1 The hierarchical navigable small world

HNSW is a popular tool for approximate nearest neighbor search in high-dimensional spaces. It is renowned for its efficient hierarchical graph structure that enables swift navigation and similarity searches, and is particularly effective at retrieving relevant documents due to its high recall rates. However, when juxtaposed with Facebook AI Similarity Search (FAISS), HNSW may demonstrate slower query times, especially

in larger dimensions. Although FAISS is designed for optimal performance and can efficiently manage large-scale datasets, HNSW's forte lies in maintaining high recall rates in dynamic settings.

Performance comparison of various models using HNSW and FAISS, presenting results based on different configurations. The comparison contains metrics such as index dimensionality, similarity threshold, precision, and recall for models including MiniLM, BERT, and RoBERTa. As an example, the BERT model employing HNSW attained a precision of 0.96 and a recall of 0.55 at a threshold of 0.9, while the same model using FAISS achieved a precision of 0.96 and a higher recall of 0.92 at a threshold of 0.7. This suggests that while HNSW can deliver high precision, FAISS may offer superior recall rates in certain circumstances.

Both libraries serve valuable purposes in different scenarios, with FAISS often being the preferred choice for speed and scalability, while HNSW is favored for its accuracy in retrieval tasks. Ultimately, the decision between HNSW and FAISS is based on the specific needs of the application, such as prioritizing speed or promoting high recall in document retrieval [53].

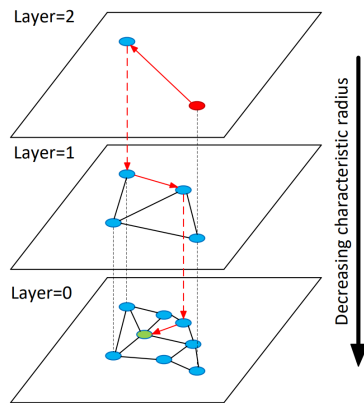


Figure 2.8 : Illustration of the HNSW [7]

The illustration 2.8 shows the main concept of HNSW algorithm through a simple three-tiered design. Each level features nodes (blue dots) with connecting paths, while red arrows indicate the search progression downward through the layers towards a target point (marked in green). The structure utilizes longer connections at higher levels and shorter ones at lower levels, creating an efficient search pattern that moves from broad to precise navigation.

2.7.2 Facebook AI similarity search

The Faiss library, offers a comprehensive set of tools and principles for efficient vector similarity search, especially in large-scale datasets. Designed to address the challenges of searching through extensive collections of embeddings, which are vector representations of various data types including text, images, and audio, Faiss has become widely adopted across industries since its open-source release in 2017.

To optimize search performance, Faiss employs several indexing techniques. These include IVF indexing, which groups database vectors during indexing to enable efficient searches by examining only a subset of clusters based on their proximity to centroids. The library also utilizes graph-based methods for rapid nearest neighbor searches and LSH to compress embeddings into compact codes. Additionally, quantization methods are implemented to reduce vector size while maintaining search accuracy, allowing for the management of large datasets.

Faiss allows for the adjustment of hyperparameters, such as nprobe for IVF, to strike a balance between speed and accuracy, focusing on identifying optimal settings for various use cases. Importantly, the library emphasizes the separation of the embedding extraction process, typically performed by neural networks, from the vector search algorithm, ensuring that the embeddings align with the intended task.

The performance of Faiss is impressive, capable of efficiently managing billions to trillions of vectors, making it suitable for large-scale applications. Its versatility is evident in various domains, including NLP for information retrieval and document clustering, data mining for organizing extensive datasets, and content moderation for detecting harmful media. The library's strong community presence and widespread adoption by major vector database companies further highlight its significance in the field.

In summary, Faiss serves as a powerful and flexible tool for vector similarity search, adept at balancing speed, memory usage, and accuracy. This makes it an invaluable resource for developers and researchers working with embeddings across various applications [54].

2.7.2.1 IndexFlatL2

In the FAISS library, IndexFlatL2 stands as one of the most basic and uncomplicated indexing methods. It functions by determining the closest neighbors among high-dimensional vectors through Euclidean distance (L2 norm) calculations. The entire vector dataset is stored in memory, and a full linear scan is conducted across all data points for each query. This method ensures precise search outcomes, making it particularly useful for smaller datasets. However, as the dataset grows, both computational complexity and memory requirements increase linearly, making it less suitable for large-scale applications.

The straightforward nature of IndexFlatL2 is evident in its absence of sophisticated optimization techniques like acceleration structures or data compression methods. This characteristic defines it as a "flat" indexing approach, emphasizing its direct and unmodified method of similarity search. While it may not be efficient for handling extensive datasets, IndexFlatL2 serves as a dependable standard for assessing the efficiency of more advanced indexing and search algorithms [8].

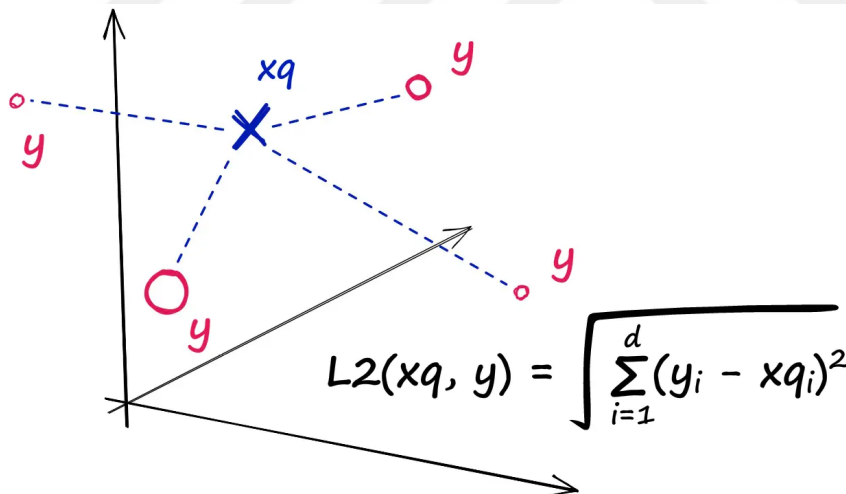


Figure 2.9 : IndexFlatL2 computes the Euclidean (L2) distance by comparing a query vector against all points in the dataset. [8]

2.7.2.2 IndexIVFFlat

The FAISS framework incorporates IndexIVFFlat index, an advanced indexing methodology designed to enhance approximate nearest neighbor searches in extensive datasets. The system utilizes Voronoi diagrams to create a sophisticated clustering

mechanism, dividing data into distinct cells around centroids. This innovative approach, based on the IVF technique, begins with a critical learning phase where vectors are grouped into predefined clusters using Euclidean distance measurements. Instead of examining every data point, the system intelligently focuses on the most pertinent clusters for each query, guided by two essential parameters: *ncells* for total cluster count and *nprobe* for search depth.

This selective approach, combined with linear searching within clusters, enables the system to process billions of vectors efficiently while optimizing memory usage. The system's adaptability through *nprobe* adjustment allows users to fine-tune the balance between speed and accuracy, making it an invaluable tool for data-intensive applications [55].

2.7.2.3 Product quantization

PQ is a method that helps to compress high-dimensional data and makes it easier to perform ANNS. The basic idea of PQ is to break a high-dimensional vector into smaller parts called sub-vectors and then quantize each one separately. This process creates a smaller version of the original vector, known as a PQ-code. A significant advantage of Product Quantization is its capability to reduce input vectors into short codes, such as 64 bits. This makes it possible to manage large collections of data, often numbering in the billions, all at once in memory. Furthermore, PQ also supports quick distance calculations; the distance between a raw vector and its corresponding PQ-code can be estimated efficiently using precomputed distance tables. This method is called asymmetric distance computation and provides a reliable approximation of the original Euclidean distance.

In addition, the algorithms and data structures involved in PQ are relatively straightforward, allowing for easy addition to other indexing systems, like inverted indices, which can boost search efficiency even further. This ease of use, along with its effectiveness, has led to PQ being used widely in various fields, including image retrieval, large-scale content-based multimedia retrieval, and deep learning, thanks to its ability to balance memory use and processing speed [55].

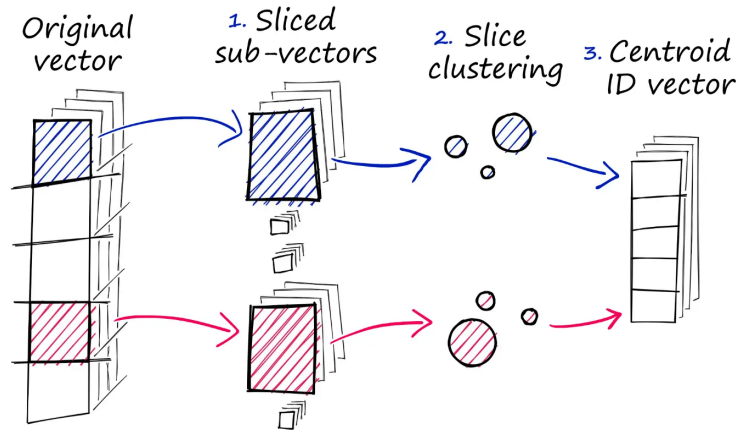


Figure 2.10 : The steps of PQ [9]

2.7.2.4 Locality-sensitive hashing

LSH is a method developed to perform efficient similarity searches in high-dimensional spaces by utilizing hash functions that group similar items into the same buckets. This technique employs specially designed hash functions that maintain distance relationships, enabling approximate nearest-neighbor searches while substantially reducing the number of necessary comparisons. LSH proves especially useful for extensive datasets, offering scalable solutions in areas such as image retrieval and document similarity analysis. The LSH Forest variation further enhances this approach by boosting query performance and removing the requirement for manual parameter adjustment [56].

2.8 Vector Search

Vector search is an innovative approach to information retrieval that utilizes vector representations of data, known as embeddings, to identify similar items based on their semantic content, rather than solely relying on traditional keyword matching. This technique converts documents, images, or other data types into high-dimensional vectors using methods such as word embeddings or deep learning models. The search process then identifies vectors closest to a query vector in the embedding space, typically using distance metrics like cosine similarity or Euclidean distance.

The advantages of vector search are substantial. It provides a more profound semantic understanding of data, yielding more relevant search results than keyword-based approaches. It can efficiently handle large volumes of unstructured data, making it

ideal for contemporary applications with extensive datasets. Its adaptability allows it to support various data types, including text and images, and it can be tailored to different domains and use cases. Furthermore, advanced algorithms and indexing techniques can significantly enhance retrieval speed and accuracy.

However, vector search is not without its challenges. The high dimensionality of the data can lead to issues such as the "curse of dimensionality," where distance metrics become less meaningful. Implementing and optimizing vector search systems can be intricate, requiring expertise in machine learning and data management. Additionally, vector search may demand significant computational resources, particularly for real-time applications or when dealing with dynamic datasets. Lastly, while ANNS methods improve speed, they may sacrifice some accuracy, which can be problematic in applications requiring precise results.

In summary, vector search represents a significant advancement in information retrieval, harnessing the power of embeddings and sophisticated algorithms to deliver more relevant and contextually appropriate search results. While it offers numerous benefits, including semantic comprehension and scalability, it also presents challenges related to complexity and resource requirements. As technology evolves, vector search is anticipated to play an increasingly important role in various applications across diverse fields [53].

2.9 Graph-Based Search

Graph-based search is a method of querying that exploits the structural characteristics of graphs to extract information from model repositories. In this technique, models are depicted as directed annotated graphs, with nodes symbolizing model components and edges representing the connections between these components. This representation encapsulates the model's structure and incorporates pertinent details as annotations.

The core function in graph-based search involves comparing a query graph—which may be an entire model or a portion thereof—to the graphs stored in the repository. This is typically accomplished through subgraph isomorphism methods, which evaluate the degree of similarity between the query graph and the repository graphs.

Furthermore, graph-based search incorporates similarity metrics that take into account the semantic associations among model components. This enables more sophisticated comparisons, allowing for the retrieval of models that are not only structurally comparable but also contextually pertinent to the query. Users can conduct searches by submitting model fragments as queries, which improves the relevance of the results by considering the relationships between elements.

In summary, graph-based search offers a robust mechanism for querying model repositories, harnessing the inherent structure and relationships within the models to facilitate efficient information retrieval [57].

2.10 Text Summarization

Text summarization is a technique that condenses lengthy texts into brief, coherent versions while preserving essential information and overall significance. This method has become increasingly crucial due to the overwhelming amount of online content available to users. A summary is characterized as a condensed version of one or more original texts that captures key points and is typically more concise than the source material. Two primary approaches to summarization exist: extractive and abstractive. The extractive method involves identifying and selecting crucial sentences or phrases from the original text based on specific criteria to create a summary. Conversely, the abstractive approach generates new sentences that encapsulate the essence of the original content, often rewording or paraphrasing rather than simply extracting information.

Researchers have developed various text summarization techniques, including statistical methods that analyze word and phrase frequency, machine learning approaches that improve summarization quality through data analysis, and graph-based methods that create text representations to identify significant sentences. Text summarization finds widespread application in numerous fields, such as search engines generating document preview snippets, news websites offering condensed article descriptions, and scientific paper summarization, where cited papers and conference details aid in identifying critical sentences.

Despite progress in the field, summarization systems face several challenges, including comprehending context and nuances within texts, managing diverse content types (including multimedia elements on web pages), and ensuring that generated summaries are coherent and maintain the original meaning. In essence, text summarization remains a vital area of research and application, aimed at enhancing information accessibility and manageability for users in an information-saturated environment [58].





3. METHODOLOGY

3.1 Data Preparation

Data cleaning and preprocessing are crucial steps in preparing web content for natural language processing tasks. This study presents a systematic approach to transforming HTML-rich blog post content stored in JSON format into clean, structured text while maintaining essential metadata and URL relationships. The methodology addresses several technical challenges inherent in web-based content, including the presence of embedded HTML elements, diverse character encodings, and complex URL structures.

The preprocessing framework encompasses multiple stages of data transformation. Initially, the system processes JSON data through UTF-8 character encoding verification, followed by the elimination of invalid control characters that might compromise data integrity. The data structure undergoes reorganization into a normalized format, implementing a dictionary-based approach for post-identification, title storage, and content management.

Central to this framework is an advanced HTML processing mechanism that preserves semantic relationships while removing structural markup. The system incorporates specialized URL handling protocols, implementing extraction and storage methodologies that maintain contextual relationships while eliminating redundant entries. This approach ensures the preservation of valuable web references while streamlining the underlying textual content.

Experimental results demonstrate the effectiveness of this preprocessing approach through comparative analysis of raw and processed datasets. The transformation process successfully normalizes content structure while maintaining semantic integrity, as evidenced by the preservation of meaningful relationships between text elements and their associated metadata. The system exhibits robust performance across varying content types and structural complexities.

The significance of this preprocessing methodology lies in its ability to generate clean, structured data suitable for subsequent natural language processing applications. The framework's adaptability to diverse input formats and its capability to handle structural inconsistencies establish its utility in content processing workflows. This research contributes to the field by presenting a reliable approach to web content preprocessing that addresses the complexities of modern digital text while preserving essential information structures.

Through empirical evaluation, the system demonstrates consistent performance in handling complex HTML structures, special character sequences, and URL patterns. The resulting cleaned dataset maintains the semantic value of the original content while eliminating unnecessary markup and standardizing text representation. This standardization facilitates improved processing efficiency in subsequent analytical tasks while ensuring the retention of crucial content relationships.

3.2 Language Model Integration in Semantic Search: A Vector and Graph-Based Approach

The implementation of this semantic search framework represents a significant advancement in multilingual information retrieval systems. The research focuses on developing a comprehensive system that leverages multiple state-of-the-art language models for efficient and accurate document retrieval. The framework's architecture carefully considers computational efficiency, scalability, and search accuracy, incorporating both vector-based and graph-based search methodologies.

At the core of this implementation are twelve carefully selected language models, each chosen for their specific strengths in multilingual text processing:

all-mpnet is a sentence transformer model built on Microsoft's MPNet and fine-tuned on over 1 billion sentence pairs. It converts sentences and paragraphs into 768 dimensional vectors for semantic tasks like information retrieval, clustering, and similarity analysis. The model can process up to 384 token sequences and be trained using tensor processing unit v3 – 8 with a contrastive learning approach.

Designed for semantic understanding across languages, it is a versatile tool for various text-processing applications [59].

pharaphrase is a sentence transformer model that maps text to 384 dimensional vector space. Based on the BERT architecture, it utilizes mean pooling and is specifically designed for multilingual applications. The model processes sequences up to 128 tokens and is optimized for various semantic tasks including clustering and semantic search across different languages. Its architecture combines a transformer model with a pooling layer that focuses on mean token representation, making it efficient for multilingual sentence embedding tasks [60].

distiluse is an advanced language model that transforms text into 512 dimensional vectors for semantic processing tasks like clustering and similarity comparison. Built on a DistilBERT architecture, it processes input text through three main components: a transformer layer (max 128 tokens), a mean pooling layer, and a dense layer with tanh activation that reduces dimensions from 768 to 512. The model is easily implemented using the sentence-transformers library and has been thoroughly evaluated through the Sentence Embeddings Benchmark framework to ensure reliable performance in real-world applications [61].

LaBSE model has made significant strides in multilingual NLP, offering support for 109 languages and delivering top-tier performance in bi-text retrieval and mining tasks.

By incorporating effective strategies such as MLM, TLM, and dual encoder translation ranking, LaBSE has managed to cut down the necessary parallel training data by up to 80%. The model's impressive capabilities are evident in its 83.7% bi-text retrieval accuracy on the Tatoeba dataset, surpassing earlier models, such as XLM-R. Remarkably, LaBSE shows proficiency even in languages without explicit training data, capitalizing on the language similarities inherent in its design. The researchers highlighted the significance of training techniques, particularly additive margin softmax, in enhancing the efficacy of the model. They also noted that while the quality of parallel data is crucial, the advantages of increasing data volume decrease when using a pre-trained model. The LaBSE model has been made publicly

accessible, providing a valuable asset for ongoing research and application in the field of multilingual NLP [62].

xlm-r-mult is a sophisticated sentence transformer model that converts text into 768-dimensional vector representations, specializing in multilingual semantic analysis and clustering tasks. Based on the XLM-RoBERTa architecture, it processes text through a transformer layer with a 128 token limit and a mean pooling layer, without requiring dimension reduction. The model offers flexibility in implementation, being accessible through both the sentence-transformers library for straightforward usage and the HuggingFace Transformers library for more detailed control over the embedding process, including custom pooling operations [63].

microsoft-m is a compact yet powerful language model developed by Microsoft, designed for efficient multilingual language understanding and generation tasks. It's a distilled version of larger transformer models, featuring 12 layers, 384 hidden dimensions, and 12 attention heads, with approximately 21M transformer parameters and 96M embedding parameters. The model uniquely combines BERT's transformer architecture with XLM-RoBERTa's tokenizer, making it particularly efficient for cross-lingual tasks. Its performance has been validated on significant benchmarks including XNLI and multilingual question answering, demonstrating its effectiveness in cross-lingual transfer learning scenarios, particularly when transferring knowledge from English to other languages [64].

intfloat-m is an advanced multilingual text embedding model, designed for sophisticated cross-lingual understanding and retrieval tasks. Developed by researchers from Microsoft, it features a robust architecture with 24 layers and 1024-dimensional embeddings. The model is built upon XLM-RoBERTa-large and has been further trained through contrastive learning with weak supervision, making it capable of handling 100 different languages. It stands out for its unique input formatting requirement, where text must be prefixed with either "query: " or "passage: " regardless of the language, and uses an average pooling strategy for generating embeddings. While it supports a wide range of languages, the model documentation notes potential performance variations for low-resource languages [40].

mbert is a foundational multilingual transformer model developed by Google, trained on Wikipedia content from 104 different languages using MLM and NSP objectives. Operating with 768-dimensional embeddings, this case-sensitive model employs a unique pre-training approach where it masks 15% of input words and predicts them in a bidirectional manner, while also learning to understand sentence relationships. The model processes text using WordPiece tokenization with a 110,000 word shared vocabulary, and handles various languages differently. For example, adding special Unicode blocks around characters in languages. During training, the model employs a balanced sampling strategy, under-sampling high-resource languages and over-sampling low-resource ones to maintain fairness across languages. It's primarily designed for fine-tuning downstream tasks such as sequence classification, token classification, and question answering, rather than text generation tasks [65].

fastText-t model is a lightweight, efficient word embedding model developed by Facebook for processing Turkish text. It generates 300-dimensional word vectors, trained on a combination of Common Crawl and Wikipedia data using the CBOW approach with position-weights, character n-grams of length 5, and a window size of 5 with 10 negative samples. The model is particularly efficient, capable of training billions of words on standard CPUs in minutes, and includes a vocabulary of 145 – 940 words. It's designed for versatile use cases including text classification, language identification, and word similarity tasks, with the ability to be compressed for mobile applications. The model uses specialized tokenization processes for different script systems and is distributed under the Creative Commons Attribution-Share-Alike License 3.0 [66].

T5-base is a versatile language model developed by Google Research with 220 million parameters and 768-dimensional embeddings, designed to handle various NLP tasks in a unified text-to-text format. Unlike BERT-style models that output only class labels or input spans, T5 can process both input and output as text strings, making it adaptable for tasks ranging from translation and summarization to question answering and classification. The model was pre-trained on the Colossal Clean Crawled Corpus using a combination of unsupervised denoising and supervised text-to-text language modeling objectives. It was trained on multiple supervised datasets covering tasks like

sentiment analysis, paraphrasing, natural language inference, and question answering. The model follows the Apache 2.0 license and supports multiple languages including English, French, Romanian, and German, demonstrating its flexibility across various NLP applications [67].

xlm-robert is a multilingual transformer model developed by Facebook AI, trained on 2.5TB of filtered CommonCrawl data spanning 100 languages. With 768 dimensional embeddings, it is based on the RoBERTa architecture and employs a MLM approach where it randomly masks 15% of input words and predicts them in a bidirectional manner. Unlike traditional sequential models or autoregressive models like GPT, this masking technique allows the model to develop a comprehensive bidirectional understanding of text. The model is primarily designed for fine-tuning downstream tasks that require whole-sentence comprehension, such as sequence classification, token classification, and question answering, rather than text generation tasks. As a base-sized model, it offers a balance between computational efficiency and performance, making it suitable for a wide range of multilingual NLP applications [68].

bge-large which is general embedding, a state-of-the-art embedding model developed by the Beijing Academy of Artificial Intelligence, designed for advanced text embedding tasks. With 1024 dimensional embeddings, it represents an improved version (v1.5) that offers a more reasonable similarity distribution compared to its predecessors. The model achieves top performance on the Massive Text Embedding Benchmark benchmark with an average score of 64.23 across 56 tasks, excelling particularly in retrieval (54.29), pair classification (87.12), and semantic textual similarity (83.11) tasks. It's primarily designed for generating dense vector representations of text that can be used for semantic search, clustering, and similarity comparison. The model requires a specific instruction prefix when processing queries for retrieval tasks, though this instruction isn't needed for processing passages or other use cases [69].

The framework implements sophisticated memory management techniques, utilizing dynamic batch processing with a batch size of 64 documents and automatic CUDA

cache clearing to optimize GPU memory usage. The embedding generation process is configured with a maximum sequence length of 512 tokens, ensuring a comprehensive capture of document semantics while maintaining computational efficiency. A particularly innovative aspect of the implementation is the graph-based search methodology, which employs a configurable similarity threshold of 0.5 and implements a weighted scoring system combining 70% direct similarity scores with 30% PageRank weights.

Search performance is enhanced through the implementation of FAISS for efficient similarity computations, supporting model-specific embedding dimensions ranging from 300 to 1024. The system incorporates careful error handling and validation mechanisms throughout the pipeline, ensuring robust performance even with varying input quality. Resource management is optimized through the lazy loading of models and cached embeddings with versioning, significantly reducing memory overhead during operation.

Result quality is maintained through comprehensive filtering and normalization processes, with configurable parameters for result count and content preview length. The implementation includes detailed logging and performance monitoring capabilities, enabling thorough analysis of search performance across different models and methodologies. This attention to both performance and quality metrics ensures that the framework can be effectively evaluated and optimized for specific use cases.

The modular design of the framework facilitates easy integration of new models and search strategies, while the comprehensive evaluation system enables detailed performance analysis across different search methodologies. The implementation demonstrates particular strength in handling multilingual content, making it especially valuable for diverse content repositories. Through careful optimization of memory usage and computational resources, the framework achieves efficient performance while maintaining search quality, establishing it as a robust solution for semantic search applications.

This research contributes significantly to the field of information retrieval by providing a detailed implementation that combines multiple state-of-the-art approaches

practically and efficiently. The framework’s ability to handle multiple languages and document types, coupled with its flexible architecture, makes it a valuable tool for both research and practical applications in semantic search.

3.3 Summarization

In this study, we developed a reference answer generation system to create evaluation inputs for assessing information retrieval system performance metrics. Initially, we experimented with various multilingual models for the summarization task, including XLM-RoBERTa, mt5-base, mt5-large, mbart-large-50-many-to-many-mmt, and mT5_multilingual_XLSum. However, these models produced summaries that lacked coherence and contextual relevance for Turkish academic content. After observing these limitations, we identified the MT5-Small Turkish summarization model as the most suitable option for our specific use case.

ozcangundes/mt5-small-turkish-summarization is a Turkish text summarization model based on Google’s Multilingual T5-small architecture. The model, containing 300 million parameters and requiring 1.2GB of storage, was fine-tuned on the Multilingual Summarization Turkish news dataset using PyTorch Lightning. The training process utilized 20,000 articles for training and 4,000 for validation from a larger dataset of 250,000 Turkish news articles. The model was trained for 10 epochs with specific parameters: a batch size of 8 and a learning rate of $10e - 4$, taking approximately 4 hours to complete. It processes input text up to 784 tokens and generates summaries up to 64 tokens in length. The system is particularly effective for summarizing Turkish news articles, as demonstrated by the provided examples in the documentation, and can be easily implemented using the Transformers library [70].

The process began with identifying 10 relevant documents `relevant_ids` that would serve as our baseline for evaluation. Using the MT5-Small Turkish summarization model, we processed these documents by combining their titles and content to generate concise summaries. The summarization pipeline was configured with specific parameters: a maximum length of 250 tokens and a minimum length of 15 tokens, along with beam search (width of 3) and a length penalty of 1.5.

These generated summaries served as reference answers, which were then used as input for various performance metrics such as Text Generation Metrics indicated strong BLEU, ROUGE, Metric for Evaluation of Translation with Explicit Ordering (METEOR), and Consensus-based Image Description Evaluation (CIDEr) scores. This approach allowed us to create a standardized set of reference points against which we could evaluate the information retrieval system's output quality and relevance. The reference answers provided a basis for the quantitative assessment of both retrieval accuracy and semantic similarity in our subsequent performance evaluations.

3.4 Performance Evaluation

A comprehensive evaluation framework was developed to assess and compare various language models and search methodologies in information retrieval systems. The framework was designed to encompass multiple metric categories, from basic performance metrics to advanced reading comprehension metrics, providing a multifaceted analysis approach. Both vector-based and graph-based search methodologies were implemented to enable direct performance comparisons across different language models. The framework was enhanced with an efficient caching mechanism to optimize computational resources and integrated with extensive visualization capabilities for result analysis. By combining traditional information retrieval evaluation methods with modern natural language processing techniques, the framework was structured to offer a holistic assessment of model performance. The results were presented through interactive visualizations and detailed reports, facilitating both broad comparisons and detailed analysis of model behaviors. This evaluation system contributes to the field by providing a standardized approach for assessing multilingual information retrieval tasks.

3.4.1 Basic metrics

In the basic metrics component of the evaluation framework, a comprehensive set of fundamental performance indicators was implemented. Precision was calculated to measure the accuracy of retrieved documents within a specified top-K range, while Recall was utilized to assess the proportion of relevant documents successfully retrieved. The F1 Score was computed as the harmonic mean of precision and recall, providing a balanced measure of the system's performance. Average Similarity scores were calculated to evaluate the semantic closeness between query and retrieved documents. The metrics were designed to handle both successful retrievals and edge cases, with appropriate error handling mechanisms. To ensure reliable comparison, the metrics were normalized and scaled appropriately. The implementation included provisions for handling multilingual content, making the basic metrics framework robust for diverse language scenarios.

Precision measures the accuracy of the positive predictions made by the model. It indicates how many of the predicted positive instances are positive [71].

$$Precision = \frac{TP}{TP + FP} \quad (3.1)$$

where

- TP : (True Positive) predicted correctly in positive instances
- FP : (False Positive) predicted incorrectly in positive instances

Recall or sensitivity, is an important metric in machine learning, especially for classification tasks. It shows how well a classifier can find all relevant instances in a dataset. Recall is calculated as the number of true positive predictions divided by the total number of actual positive instances [72]. The formula for the recall is:

$$Recall = \frac{TP}{TP + FN} \quad (3.2)$$

where

- *FN* : (False Negatives) refers to the number of actual positive instances that were incorrectly classified as negative.

F1 Score is the harmonic mean of precision and recall. It provides a single score that balances both metrics, making it useful when you need to find an optimal balance between precision and recall [73].

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.3)$$

Similarity metric is a way to measure how similar two datasets, documents, or examples are. There are different types of similarity metrics. One type is distributional similarity metrics like KL-divergence, which looks at how different probability distributions are. Another type is embedding-based metrics, such as the Magnitude-Aware Uncertainty Variance Estimation score and cosine similarity, which compare the vector forms of text. Language model perplexity is also used to see how well a model can predict a sample, with lower perplexity showing a closer match to the training data. The similarity metrics relate to how well models perform on multilingual datasets, they do not always predict success across all tests. This indicates a complicated link between similarity and the performance of language models [74].



Several notable patterns emerged in the analysis of basic performance metrics across different models. The LaBSE model consistently demonstrated superior performance across all metrics, achieving the highest scores of 0.7 for Precision@K and Recall@K in both vector and graph-based approaches. Its F1 Score of 0.7 further reinforced this dominance, indicating balanced precision and recall performance.

When comparing vector and graph-based approaches, a graph-based search showed generally higher consistency across models, particularly evident in the performance of distiluse and paraphrase models, which achieved scores of 0.5 – 0.6 across metrics. The vector-based approach showed more variation in performance, with models like all-mpnet and paraphrase achieving moderate scores (0.4 – 0.5). In contrast, others like xlm-robert and microsoft-m performed notably poorly with scores around 0.1.

In terms of Average Similarity, an interesting divergence was observed between vector and graph-based approaches. The vector search showed extremely high similarity scores for the intfloat-m model (317.5), while graph-based search demonstrated more normalized similarity scores across models, with several models achieving scores between 0.6 – 0.7.

Overall Assessment: LaBSE emerged as the most robust and consistent performer across all basic metrics in both search approaches. Graph-based search generally demonstrated more stable and consistent performance across different models compared to vector-based search. The poorest performing models were consistently t5-base and xlm-robert, showing minimal effectiveness across all metrics. This suggests that while model selection is crucial, the choice of search methodology (graph vs. vector) can significantly impact retrieval performance, with graph-based approaches potentially offering more reliable results across a broader range of models.

3.4.2 Performance metrics

In the performance metrics component of the evaluation framework, several key efficiency and performance indicators were implemented. Response Time was measured to evaluate the computational efficiency and speed of retrieval operations for each query. Mean Reciprocal Rank (MRR) was calculated to assess the ranking

quality of the retrieved results, focusing on the position of the first relevant document. Mean Average Precision (MAP) was implemented to evaluate the ranking accuracy across all relevant documents, providing insights into the overall retrieval effectiveness. Coverage metrics were developed to measure the system's ability to retrieve relevant documents from the entire collection. The metrics were designed with robust error-handling mechanisms to ensure accurate measurements even in edge cases. Timing mechanisms were implemented with high precision to capture even subtle differences in response times. The framework included automated performance logging and monitoring capabilities to track system behavior under different load conditions and query complexities.

Response Time is an important measure of performance. It indicates how long it takes to receive the first response after a request is made. Several factors affect this time, including how the component is built, how long required services take to execute, the platform on which it runs, usage patterns, and competition for resources. Various methods can measure and predict response times, such as platform-independent assessments and formal performance standards. In performance modeling, the total response time for a component assembly is the sum of the response times of each component. Improving response time is essential for fulfilling performance needs and providing a good user experience [75].

Coverage systems has the ability to suggest many items, especially those users haven't rated yet. This improves how well the system works by recommending things users might like. By looking at items that haven't been rated, systems can better understand how similar users can help in predicting ratings for those items. Enhancing coverage is crucial for users to find new items. Using methods that weigh user rating frequency can improve both coverage and prediction accuracy. More coverage results in a broader and more effective recommendation experience [76].

MRR is a widely utilized metric for assessing recommendation system effectiveness, especially in the field of information retrieval. This measure quantifies how well a system places relevant items at the top of its ranking list. MRR is calculated by averaging the reciprocal ranks of the first relevant item for a group of queries,

concentrating exclusively on the position of the initial relevant result when multiple items are presented. A superior MRR score suggests better system performance, indicating that relevant items are more frequently positioned higher in the rankings. An MRR of 1.0 demonstrates that the first relevant item was consistently found in the top position across all queries, while a score approaching 0 indicates subpar performance. This metric proves particularly valuable in situations where the primary objective is to swiftly deliver the most pertinent items to users, making it an essential tool for evaluating recommendation systems [77].

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \quad (3.4)$$

where

- Q : A set of queries for which the recommendation system is evaluated.
- rank_i : The position of the first relevant item in the ranked list returned by the recommendation system for the i -th query.

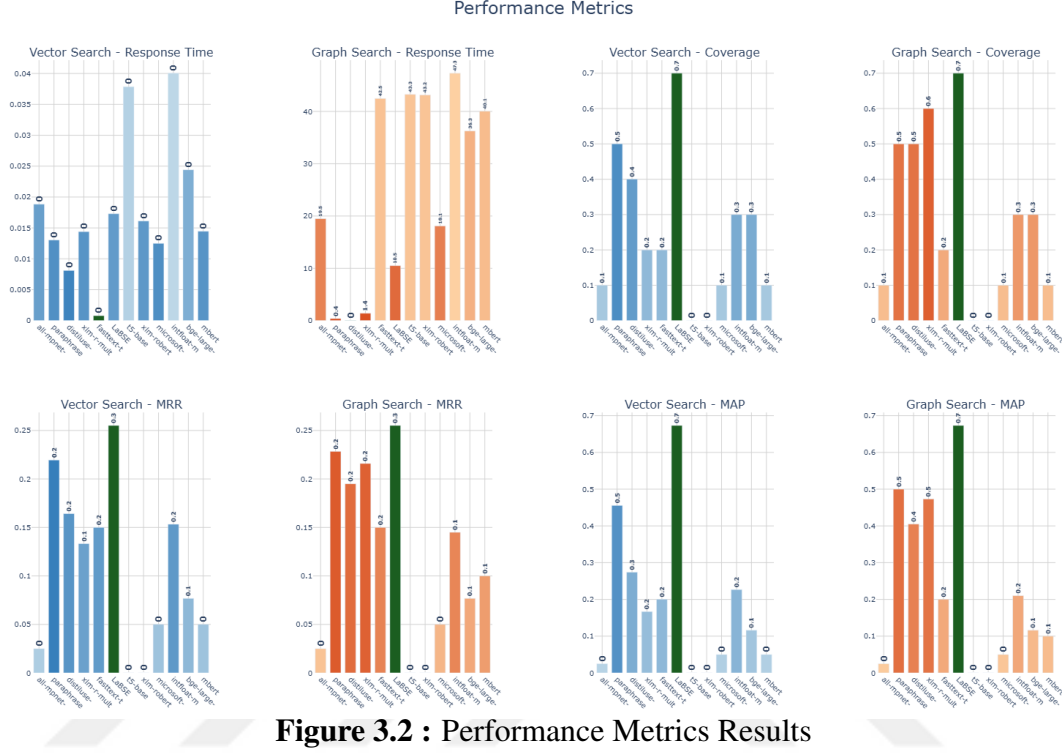
MAP is a crucial metric for assessing object detection algorithm performance. This measure consolidates a model's precision and recall across various classes and thresholds by calculating the mean of Average Precision (AP) scores for each object category. AP is calculated from precision and recall values at different confidence score thresholds, typically represented as the area beneath the precision-recall curve.

MAP's importance stems from its capacity to evaluate model performance across multiple classes, making it particularly valuable for detecting diverse object types. In essence, MAP offers a comprehensive assessment of an object detection model's efficacy in identifying and positioning objects, serving as a vital metric for comparing and benchmarking different models [78].

$$mAP = \frac{1}{j} \sum_{m=1}^j AP_{C_m} \quad (3.5)$$

where

- j : The number of object categories.
- AP_{C_m} : Average Precision for category C_m , which measures the precision and recall for that specific category.



In the analysis of performance metrics, several distinct patterns emerged across different models and search approaches. For Response Time, vector-based search demonstrated significantly faster performance overall, with most models operating in the 0.01 – 0.04 second range. In contrast, graph-based search showed notably higher response times, particularly for models like mbert and bge-large (40+ seconds). The fastest model in vector search was fasttext-t, while distiluse showed the best response time in graph search.

For Coverage metrics, LaBSE consistently demonstrated superior performance with a score of 0.7 in both vector and graph-based approaches. Graph-based search showed better overall coverage, with models like distiluse and paraphrase achieving scores of 0.5 – 0.6, compared to their lower performance in vector search. T5-base and xlm-robert performed poorly in both approaches, showing zero coverage.

Regarding MRR, LaBSE again led with a score of 0.3 across both search methods. Paraphrase and distiluse models performed stronger in graph search (0.2) than their vector search counterparts. Models like t5-base and xlm-robert consistently performed poorly with scores of 0.

The MAP results followed a similar pattern, with LaBSE achieving the highest score of 0.7 in both approaches. Graph search showed more consistent performance across different models, particularly for paraphrase (0.5) and distiluse (0.4), while vector search showed more variable results.

Overall Assessment: LaBSE emerged as the most consistent and high-performing model across all performance metrics in both search approaches. Graph-based search generally showed more stable but slower performance across models, while vector-based search offered significantly faster response times but with more variable performance across other metrics. The poorest performing models were consistently t5-base and xlm-robert. This suggests that while model selection significantly impacts performance, there's a clear trade-off between speed (vector search) and consistency (graph search) in the search methodology choice.

3.4.3 Similarity metrics

The similarity metrics component implemented comprehensive measures to evaluate semantic relationships between queries and retrieved documents. Manhattan Distance was calculated to measure the absolute distance between document vectors in the embedding space. Rank Correlation was computed to assess the consistency of document rankings based on their semantic similarities. The implementation included mechanisms for handling high-dimensional embeddings and normalization procedures to ensure comparable similarity scores across different embedding spaces. Special attention was given to handling multilingual content by implementing language-agnostic similarity calculations.

Manhattan distance, also referred to as the L1 norm or Taxicab distance, calculates the separation between two points in a grid-based system by adding the absolute differences of their coordinates. This metric is especially beneficial in situations

where movement is limited to perpendicular paths, making it relevant in areas such as urban design, robotics, and computer graphics. As dimensionality increases, the mean and variance of the Manhattan distance exhibit predictable patterns, which is essential for comprehending data distributions in high-dimensional spaces. Although it is a useful measure, it may occasionally introduce inaccuracies when estimating the Euclidean distance. The Manhattan distance finds widespread application in data analysis, machine learning, and optimization challenges, offering a practical method for quantifying spatial relationships in multidimensional contexts [79].

The formula for the Manhattan distance:

$$D_M(P, Q) = \sum_{i=1}^n |x_i - y_i| \quad (3.6)$$

where

- $D_M(P, Q)$: This represents the Manhattan distance between the two points P and Q.
- P : This is the first point in n-dimensional space, represented as $P = (x_1, x_2, \dots, x_n)$, where each x_i is a coordinate of point P in the $i - th$ dimension.
- Q : This is the second point in n-dimensional space, represented as $Q = (y_1, y_2, \dots, y_n)$, where each y_i is a coordinate of point Q in the $i - th$ dimension.
- n : This denotes the number of dimensions in the space. The distance is calculated across all dimensions from 1 to n .
- $|x_i - y_i|$: This represents the absolute difference between the coordinates of points P and Q in the $i - th$ dimension. The absolute value ensures that the distance is non-negative, regardless of the order of the coordinates.

Rank correlation serves as a statistical tool for assessing the intensity and orientation of the relationship between two ranked variables, making it particularly suitable for ordinal data analysis. The primary rank correlation coefficients are Kendall's Tau and Spearman's Rank Correlation Coefficient, which find extensive use in disciplines such as psychology, economics, and social sciences for examining ordinal data relationships, including survey response analysis.

Traditional rank correlation methods like Kendall's Tau and Spearman's Rho face challenges when dealing with incomplete data, leading to recent investigations into their performance under such circumstances. While these coefficients demonstrate resilience to outliers and do not assume linear relationships, they may not be as effective as alternative methods in capturing relationship strength when data follows a normal distribution. Nevertheless, rank correlation remains an essential instrument for analyzing ordinal relationships, with ongoing studies aimed at improving its applicability in scenarios involving incomplete data [80].

$$\text{Kendall's Tau } (\tau): = \frac{n_c - n_d}{\frac{1}{2}n(n-1)} \quad (3.7)$$

where

- n_c is the number of concordant pairs (pairs where the ranks for both variables agree),
- n_d is the number of discordant pairs (pairs where the ranks for both variables disagree),
- n is the total number of observations.

$$\text{Spearman's Rank Correlation Coefficient } (\rho): = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)} \quad (3.8)$$

where

- d_i is the difference between the ranks of each observation,
- n is the number of observations.

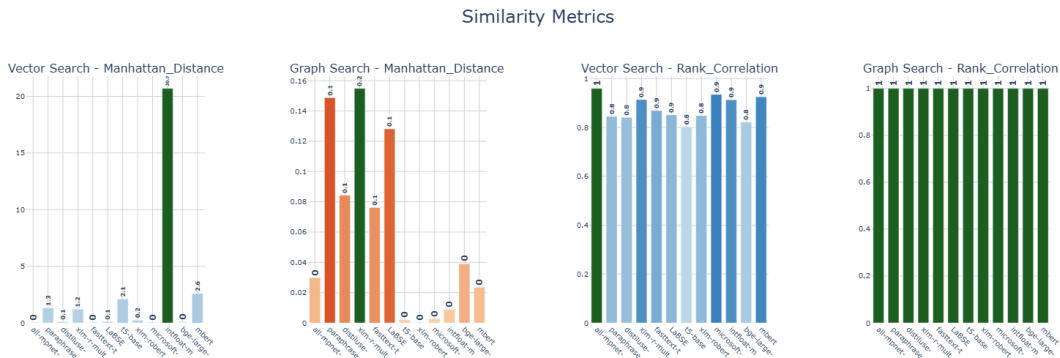


Figure 3.3 : Similarity Metrics Results

In the analysis of similarity metrics, distinct patterns emerged across different models and search methodologies. For Manhattan Distance, vector-based search showed significantly higher magnitudes, with intfloat-m achieving the highest score of 20.7, while other models like xlm-r-mult and paraphrase showed moderate distances (1.2 – 2.1). In contrast, graph-based search demonstrated much lower Manhattan distances overall, with xlm-r-mult showing the highest score of 0.16, followed by LaBSE and paraphrase (0.1 – 0.13).

The Rank Correlation analysis revealed interesting differences between the two search approaches. In vector search, while all models performed relatively well with scores above 0.8, there was some variation among models. All-mpnet achieved perfect correlation (1.0), while models like microsoft-m, intfloat-m, and mbert showed strong correlations (0.9). In contrast, graph-based search demonstrated remarkable consistency, with all models achieving perfect rank correlation (1.0), suggesting that graph-based search maintains more consistent ranking patterns regardless of the model used.

Overall Assessment: The similarity metrics analysis reveals a fundamental difference in how vector and graph-based searches handle semantic relationships. While vector search shows higher magnitude differences in Manhattan Distance, potentially indicating more distinctive semantic separations, graph search demonstrates more normalized and consistent distance measurements. The perfect rank correlation across all models in graph search suggests it might be more reliable for maintaining consistent document rankings. Intfloat-m and xlm-r-mult emerged as notable performers in distance metrics, while all-mpnet excelled in ranking consistency for vector search. This suggests that the choice between vector and graph search might depend on whether the application prioritizes distinctive semantic separation (vector) or consistent ranking behavior (graph).

3.4.4 Clustering metrics

In the clustering metrics component, systematic measures were implemented to evaluate the quality of document groupings and their semantic relationships. The Silhouette Score was calculated to assess the cohesion and separation of document

clusters, providing insights into the quality of semantic groupings. Davies-Bouldin Score was implemented to evaluate the average similarity between clusters while considering the internal scatter of the clusters. The metrics were designed to handle edge cases robustly, including situations with minimal cluster separation or outlier documents. Specialized normalization techniques were applied to ensure consistent clustering evaluation across different document collections.

Silhouette Score is a measurement tool used to assess the effectiveness of clustering algorithms. It evaluates the separation between clusters and the similarity of points within each cluster. The score ranges from -1 to 1 , with values closer to $+1$ indicating well-defined clusters where points are significantly distant from neighboring clusters. A score of 0 suggests that points are situated near or on the boundary between two adjacent clusters, while a negative score implies potential misclassification, as points may be closer to a different cluster than their assigned one.

A higher silhouette coefficient signifies superior clustering performance and reduced functional overlap, demonstrating the efficacy of the operations conducted on the tool descriptions. This evaluation method helps ensure that tools are distinct and their functionalities are clearly defined, thereby enhancing tool selection and usage awareness in large language models [81]. The silhouette score for a single point is calculated using the following formula:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \quad (3.9)$$

where

- $a(i)$ is the average distance between the point i and all other points in the same cluster
- $b(i)$ is the average distance between the point i and all points in the nearest cluster

In the **Cluster Quality** part Davies-Bouldin Index (DBI) is calculated. DBI serves as a measure for assessing clustering algorithm effectiveness, emphasizing the maximization of distances between clusters while minimizing distances within

clusters. It evaluates the proximity of points within a cluster and the separation between different clusters. A superior clustering quality is indicated by a lower DBI score, signifying compact and distinctly separated clusters, whereas a higher score suggests inferior clustering. Due to its straightforward nature and efficacy in condensing clustering performance, the DBI finds widespread application in areas such as machine learning and data mining. The DBI measures how similar each cluster is to its closest cluster. This similarity is found by comparing the total distances within clusters to the distances between different clusters [82].

$$DBI = \frac{1}{N} \sum_{i=1}^N \max_{j \neq i} \left(\frac{S_i + S_j}{d_{ij}} \right) \quad (3.10)$$

where

- S_i and S_j are the average distances of points in clusters i and j to their respective centroids, and d_{ij} is the distance between the centroids of clusters i and j

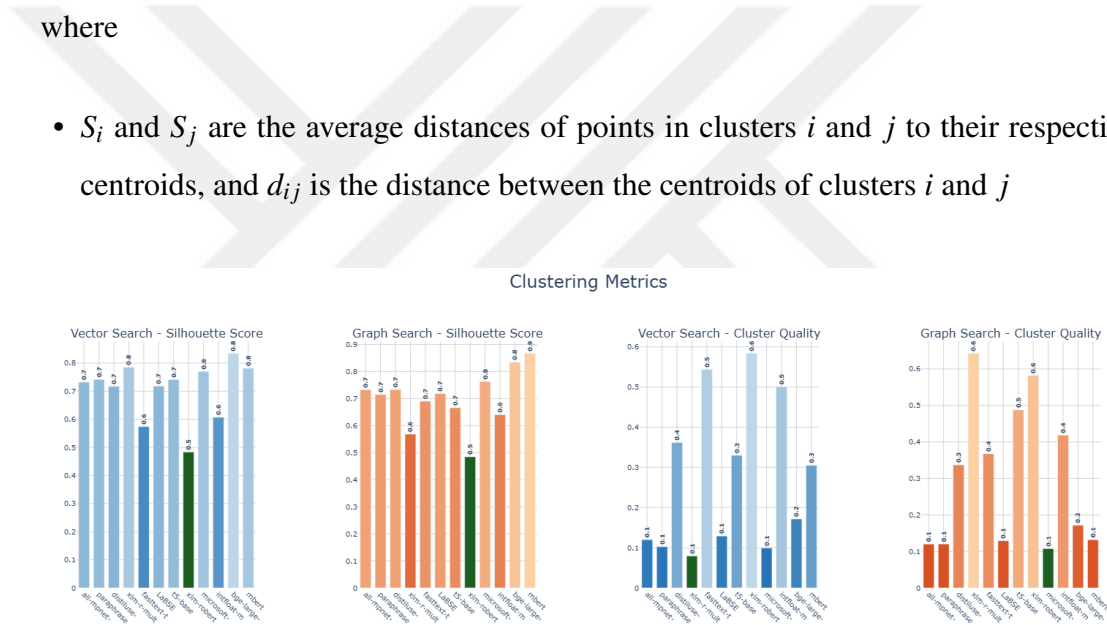


Figure 3.4 : Clustering Metrics Results

In the analysis of clustering metrics, significant variations were observed across different models and search approaches. For Silhouette Score, which measures cluster cohesion and separation, both vector and graph searches showed generally strong performance. In vector search, bge-large achieved the highest score of 0.8, followed closely by xlm-r-mult at 0.8. Graph search demonstrated even stronger clustering performance, with mbert achieving the highest score of 0.9, followed by bge-large at 0.8. Notably, microsoft-m maintained consistent performance (0.6 – 0.8) across both search approaches.

The Cluster Quality metric revealed different patterns between the two search methodologies. In vector search, t5-base showed the highest quality score of 0.6, followed by fasttext-t and bge-large at 0.5. Graph search demonstrated similar peak performance with xlm-r-mult and microsoft-m reaching 0.6, but showed more variation across models. Both approaches showed some models performing poorly, with scores around 0.1, particularly for all-mpnet and paraphrase.

Overall Assessment: The clustering metrics analysis reveals that while both search approaches can achieve high-quality clustering, they excel in different aspects. Graph search generally produced better Silhouette Scores, indicating more distinct and well-separated clusters, while vector search showed more consistent Cluster Quality scores across models. Mbert and bge-large emerged as particularly strong performers in cluster separation, while t5-base and xlm-r-mult demonstrated superior cluster quality. The significant variation in performance across models suggests that model selection is crucial for clustering tasks, with different models optimizing either cluster separation or overall quality.

3.4.5 Diversity metrics

In the diversity metrics component, multiple measures were implemented to assess the variety and coverage of retrieved results. Intra-List Similarity (ILS) Score was calculated to evaluate the diversity within the retrieved document sets. Embedding Diversity metrics were developed to measure the spread of documents in the semantic space. Feature Variance Score and Pairwise Distance metrics were implemented to assess the distributional characteristics of retrieved documents. Uniqueness Score was calculated to evaluate the distinctiveness of individual documents within the result set. A Normalized Diversity metric was developed to provide a standardized measure of result set diversity. The implementation included mechanisms for handling varying result set sizes and different embedding dimensionalities.

ILS examines the connections between various labels, including their hierarchical relationships and frequency of co-occurrence. This concept significantly enhances the accuracy of label propagation by enabling algorithms to exploit the commonalities among labels that often appear. A matrix derived from hidden concepts can be used

to quantitatively represent this similarity, facilitating a deeper understanding of how closely related different labels are. While conventional approaches typically rely on overall image similarity, which may not capture localized label associations, the suggested method emphasizes local label similarities to achieve superior annotation results. By integrating ILS, the algorithm improves its capacity to effectively propagate labels, resulting in greater precision when identifying and ranking pertinent labels. [83]

Embedding Diversity is a metric calculated by finding the variance of each feature in the embeddings matrix along the columns and then taking their mean, resulting in a single score that indicates the level of diversity among embeddings. A high score suggests significant diversity among embeddings while a low score indicates similarity, making this metric crucial for measuring how well embeddings represent different features in various contexts such as language models, feature extraction quality assessment, model diversity measurement, and overfitting/underfitting analysis. The score effectively provides insights into how well a model captures and represents various features in its embedding space, with higher scores indicating better feature representation and lower scores suggesting more uniform or similar embeddings.

Pairwise distance, serves as a crucial mechanism for maintaining diversity in generative models by measuring and comparing the spatial relationships between pairs of points across both input and output spaces. The method operates by calculating normalized distances between point pairs, using modified distance calculations tailored to specific tasks, and employing a normalization process that prevents arbitrary scaling while preserving meaningful relationships between samples. The innovation lies in its training approach, where the normalizer remains constant during gradient computation, enabling the model to explore new spaces while maintaining stable point relationships. This strategy effectively prevents mode collapse and ensures smooth interpolation between points, leading to more reliable and diverse outputs that maintain their relative spatial relationships throughout the generation process [84].

Feature Variance Score quantifies feature variability by computing individual feature variances and aggregating them into a unified metric. This calculation yields insights into the information content of features - a higher score suggests rich, diverse feature information due to substantial variation in values, while a lower score indicates more uniform or redundant features with minimal variation.

Although this metric shares similarities with Embedding Diversity, it differs in its focus: Feature Variance Score analyzes each feature's independent variability, whereas Embedding Diversity examines the collective variation patterns across all features.

Uniqueness Score is a metric that measures how distinct embeddings are from each other by calculating the average Euclidean distance between vector pairs, while also evaluating match quality by comparing the best match score to the next best one outside an exclusion window.

Higher distances indicate greater distinctiveness, while lower distances show similarity. The score equals zero when only one embedding exists since comparison requires at least two data points. This metric helps filter unreliable matches through thresholding and can be automatically optimized for better precision and recall [85].

Normalized Diversity value is calculated by dividing the sum of three different diversity metrics (embedding diversity, pairwise distance score, and feature variance score) by three times the maximum value among these metrics. This calculation results in a normalized value between 0 and 1, allowing diversity metrics to be comparable across different scenarios.



Figure 3.5 : Diversity Metrics Results

In the analysis of diversity metrics across different models and search methodologies, several distinctive patterns emerged. In ILS Score evaluation, vector search showed significantly higher scores, with intfloat-m achieving a remarkable score of 17, while mbert followed with 3.5. Graph search demonstrated more moderate ILS Scores, with distiluse and paraphrase models showing consistent performance around 0.05 – 0.06.

For Embedding Diversity, vector search showed exceptional performance with intfloat-m reaching a score of 300, significantly outperforming other models. Graph search demonstrated much lower but more consistent embedding diversity scores, with most active models showing scores around 0.002 – 0.003.

Pairwise Distance metrics revealed similar patterns to ILS Score, with intfloat-m dominating in vector search (around 20) and showing more normalized scores in graph search. Feature Variance analysis showed intfloat-m achieving the highest score in vector search (approximately 300), while graph search maintained more consistent, lower variances across models.

The Uniqueness Score metric showed intfloat-m achieving the highest score (around 22) in vector search, while graph search demonstrated more moderate scores with distiluse and paraphrase models leading (0.06 – 0.07). For Normalized Diversity, both search approaches showed more consistent performance across models, with scores ranging from 0.3 to 0.9 in vector search and maintaining around 0.3 – 0.4 in graph search.

Overall Assessment: The diversity metrics analysis reveals a clear distinction between vector and graph-based search approaches. Vector search consistently produced higher magnitude scores with greater variation between models, particularly evident in intfloat-m’s exceptional performance across all diversity metrics. Graph search, while showing lower absolute scores, demonstrated more consistent and stable diversity measurements across different models. This suggests that vector search might be more suitable for applications requiring high diversity distinctiveness, while graph search could be preferred for applications requiring stable and consistent diversity measurements across different models. Notably, intfloat-m emerged as the standout performer in vector search, while distiluse and paraphrase models showed more reliable performance in graph search across all diversity metrics.

3.4.6 Text generation metrics

In the text generation metrics component, comprehensive evaluation measures were implemented to assess the quality of generated textual content. BLEU scores were calculated to evaluate the precision of n-gram matches between retrieved and reference texts. ROUGE metrics (ROUGE-1, ROUGE-2, ROUGE-L) were implemented to assess content overlap and sequence matching. METEOR score was computed to evaluate semantic similarity while accounting for synonyms and paraphrases. CIDEr

score was implemented to measure consensus in content evaluation. Perplexity metrics were developed to assess the fluency and naturalness of the retrieved text content.

BLEU score is a commonly used metric for assessing the quality of machine-generated text in translation and other natural language processing tasks. It compares the output text to one or more reference texts by examining the overlap of n-grams, which are consecutive sequences of n items. Typically, BLEU considers unigrams, bigrams, trigrams, and four-grams. The score is calculated by determining the precision for each n-gram level, which is the ratio of matching n-grams in the generated text to the total number of n-grams in that text.

To discourage the production of excessively short outputs that could achieve high precision by matching only a few words, BLEU employs a brevity penalty. This penalty is applied when the generated text is shorter than the reference text. The final BLEU score is derived by combining the precision scores for various n-gram levels, usually through the geometric mean, and then applying the brevity penalty.

BLEU's simplicity and ease of computation make it a popular choice for evaluating text generation tasks. Its consideration of multiple n-gram levels allows for a more detailed assessment of text similarity. However, BLEU has limitations. It emphasizes precision, which can result in high scores for incomplete or irrelevant text. It doesn't account for recall, potentially missing important information from the reference texts. The brevity penalty may disproportionately affect shorter texts, and BLEU can't consider semantic meaning or context, potentially giving high scores to syntactically correct but semantically nonsensical outputs. While BLEU is a valuable evaluation tool, researchers should be aware of its shortcomings and consider using it in conjunction with other metrics for a more comprehensive evaluation of model performance.

In mathematical terms, n-gram precision is calculated as the number of matching n-grams divided by the total n-grams in the generated text. The brevity penalty equals 1 if the generated text is longer than the reference; otherwise, it's computed using an exponential function based on the length ratio. The final BLEU score is the product of the brevity penalty and the exponential of the average logarithmic precision across n-gram levels [86];

$$\text{Precision}_n = \frac{\text{Number of matching n-grams}}{\text{Total number of n-grams in generated text}} \quad (3.11)$$

$$\text{Brevity Penalty (BP)} = \begin{cases} 1 & \text{if } \text{len}(\text{generated}) > \text{len}(\text{reference}) \\ e^{\left(1 - \frac{\text{len}(\text{reference})}{\text{len}(\text{generated})}\right)} & \text{if } \text{len}(\text{generated}) \leq \text{len}(\text{reference}) \end{cases} \quad (3.12)$$

$$\text{BLEU} = \text{BP} \times \exp \left(\frac{1}{N} \sum_{n=1}^N \log(\text{Precision}_n) \right) \quad (3.13)$$

where

- number of matching n-grams refers to the count of n-grams in the generated text that also appears in the reference text.
- BP is a BLEU score component that penalizes translations that are shorter than their reference texts, encouraging appropriate output lengths.
- $\text{len}(\text{generated})$ and $\text{len}(\text{reference})$ represent the lengths in terms of the number of tokens that are generated and reference texts, respectively.
- N is the maximum n – gram length considered (usually 4 for BLEU–4).
- The term $\sum_{n=1}^N \log(\text{Precision}_n)$ calculates the logarithm of the precision for each n-gram level, which is then averaged by dividing by N .
- The exponential function is applied to this average to ensure that the final score remains within a reasonable range.

ROUGE metrics serve as essential tools for evaluating the quality of computer-generated text summaries. These metrics operate by analyzing similarities between generated summaries and reference texts in multiple ways. The most straightforward versions are ROUGE-1 and ROUGE-2, which examine matches of individual words and consecutive word pairs respectively, while more advanced variants like ROUGE-SU4 incorporate flexibility by allowing gaps between matched words.

Many researchers value ROUGE because it's been proven to align well with how humans judge summary quality. The three main versions - ROUGE-1, ROUGE-2, and ROUGE-SU4 - have become standard tools in summarization research and practical applications.

ROUGE-L measures the similarity between texts by finding the longest common subsequence between the reference and candidate summaries, where words must appear in the same order but need not be consecutive, allowing for a flexible way to capture sentence-level structure and word order similarities. Unlike other ROUGE variants that use fixed n-gram sizes, ROUGE-L automatically finds the longest matching sequences of words between texts, making it particularly effective for evaluating summaries that maintain similar content structure but may use different phrasing.

Despite its widespread use, ROUGE has notable drawbacks. It mainly looks at exact word matches, which can be problematic when evaluating summaries that convey the same meaning using different words. This is particularly challenging when assessing abstractive summaries that rephrase original content. Another issue is that researchers often choose ROUGE variants without clear justification, leading to inconsistent evaluation results.

To address these challenges, researchers suggest improving ROUGE by adding semantic analysis capabilities. This enhancement would help the metric understand deeper meaning relationships between texts, rather than just matching identical words. Such improvements would make ROUGE more effective for evaluating various types of summaries [87].

$$\text{ROUGE-N} = \frac{\text{Number of overlapping n-grams}}{\text{Total number of n-grams in the reference summary}} \quad (3.14)$$

METEOR metric is a popular tool for evaluating machine-generated text, especially in machine translation and image captioning. Unlike earlier tools like BLEU, which mainly focus on matching n-grams, METEOR offers a broader evaluation by including several important features.

A key advantage of METEOR is its word-matching method. It assesses generated captions against reference captions by looking for exact matches as well as stemming and synonyms. This approach captures semantic similarities that simpler metrics might miss. METEOR also uses both precision and recall in its scoring. Precision measures the matched words in the candidate caption, while recall looks at matched words in the reference captions. This balance gives a more detailed score that reflects the quality of the text.

Another significant aspect of METEOR is its penalty for fragmentation. The metric penalizes candidates with many short matches that are spread out across reference captions, promoting longer contiguous matches. This encourages coherence and fluency in the generated text, which is essential for natural language.

Additionally, METEOR tends to align better with human evaluations compared to other metrics. This makes it a more trustworthy option for assessing the quality of the generated text, as it closely matches human views on good captions or translations. Overall, METEOR marks an important improvement in evaluating machine-generated text by considering meaning, word order, and text coherence, thereby providing a more accurate measure of quality for image captioning and translation tasks [88].

$$F = \frac{(1 + \beta^2) \cdot P \cdot R}{(\beta^2 \cdot P) + R} \quad (3.15)$$

$$\text{METEOR} = \frac{1}{P + R} \cdot F \cdot M \quad (3.16)$$

where

- P is the precision, calculated as the number of matched words divided by the total number of words in the candidate caption.
- R is the recall, calculated as the number of matched words divided by the total number of words in the reference captions.
- F which is a harmonic mean of precision and recall
- β is a parameter that determines the weight of recall in the F measure

- M is a measure of the alignment of the matched words, which includes a penalty for fragmentation.

CIDEr score is a popular metric created by Vedantam and colleagues to assess image description systems. It measures how similar candidate descriptions are to reference descriptions by using cosine similarity. Typically, it calculates for n-gram values ranging from 1 to 4. Improved versions, CIDEr-D and CIDEr-R, incorporate Gaussian penalties and allow more flexible evaluations for longer sentences, respectively.

The main strengths of CIDEr include capturing semantic accuracy, showing strong agreement with human ratings, and handling synonyms well. On the downside, it has a high computational cost, needs multiple references, and does not effectively assess grammatical accuracy. CIDEr has become a standard metric, particularly for developing and comparing image description systems. It is often used alongside metrics like BLEU and METEOR to provide a fuller evaluation. This metric is useful as it tackles natural language description challenges by focusing on semantic content and consensus among human references, making it a reliable tool for judging the quality of automatically produced image descriptions [89].

Perplexity is a key measure for evaluating language models. It indicates how well a model predicts a sequence of words. Mathematically, perplexity is calculated by taking the exponent of the average negative log-likelihood of a sequence. For a sequence of N words, the log-likelihood is found by adding the logarithms of the probabilities assigned to each word by the model. This total is then averaged and used to determine perplexity.

In practical terms, perplexity represents the number of choices the model faces when selecting the next word. A perplexity of 10 suggests the model feels as uncertain as if it had to choose from 10 equally likely options. Lower perplexity means the model is more confident in its predictions, indicating a better grasp of language structure. Higher perplexity suggests the model struggles to predict the next word, which shows weaker performance.

Perplexity is commonly used to evaluate and compare various language models. It provides a numerical way to assess predictive ability. Monitoring perplexity during

training helps track progress; a drop in perplexity usually indicates effective learning. However, perplexity can vary by dataset, meaning a model might excel with one dataset but not another. Comparisons should occur within the same dataset for accuracy.



Several distinct patterns emerged when analyzing text generation metrics across different models and search approaches. For BLEU scores, LaBSE demonstrated superior performance with a score of 0.2 in both vector and graph searches. Paraphrase and xlm-r-mult models consistently performed around 0.2 in vector search, while bge-large achieved 0.2 in graph search.

In ROUGE metrics (ROUGE-1, ROUGE-2, ROUGE-L), LaBSE consistently maintained high performance across both search approaches, achieving scores of 0.3 – 0.35. The graph search showed robust results with bge-large and intfloat-m models reaching comparable scores (0.2 – 0.3) across all ROUGE variants.

METEOR Score analysis revealed strong performance from LaBSE (0.35) in both search approaches, with distiluse showing competitive performance in graph search (0.34). Intfloat-m and bge-large maintained consistent scores around 0.3 across both search methodologies.

CIDEr Score evaluation showed LaBSE leading with 0.08 in both vector and graph searches, while other models like paraphrase and distiluse demonstrated moderate performance (0.05 – 0.07). Some models, particularly t5-base and xlm-robert, consistently showed lower performance across both search approaches.

Perplexity measurements showed relatively consistent scores across models (1.0 – 1.3), with distiluse showing slightly higher perplexity in vector search (1.3) and several models maintaining scores around 1.1 – 1.2 in graph search.

Overall Assessment: LaBSE emerged as the consistently top-performing model across all text generation metrics in both search approaches. Graph-based search generally showed more balanced performance across different models, particularly evident in ROUGE and METEOR scores. The analysis reveals that while vector search often produced more varied results, graph search maintained more consistent performance across different models. T5-base and xlm-robert consistently underperformed across most metrics, suggesting their limited suitability for text generation tasks. This comprehensive evaluation suggests that model selection

significantly impacts text generation quality, with LaBSE offering the most reliable performance across all metrics and search methodologies.

3.4.7 Contextual metrics

In the contextual metrics component of the evaluation framework, several sophisticated measures were implemented to assess the semantic understanding and relevance of retrieved results. Query Relevance was calculated to evaluate how well retrieved documents aligned with the original query intent and requirements. Semantic Coherence metrics were developed to assess the logical consistency and flow between retrieved documents. Topic Consistency was implemented to measure how well the retrieved documents maintained focus on the central theme. Reference Alignment was computed to evaluate the similarity between retrieved content and reference documents. Special attention was given to context preservation across different languages and domains, with robust handling of semantic drift. The implementation included mechanisms for evaluating both explicit and implicit contextual relationships, ensuring a comprehensive assessment of contextual understanding. Particular emphasis was placed on maintaining contextual relevance across document boundaries and ensuring semantic consistency throughout result sets.

In the field of Learning to Rank, **query relevance** techniques are essential for enhancing ranking systems. These methods offer several benefits, including improved user satisfaction by presenting the most pertinent items, thus increasing the chances of users finding desired information. Moreover, query relevance approaches can be customized to suit individual preferences, leading to more personalized search outcomes. They also provide clear performance indicators, such as precision, recall, and Normalized Discounted Cumulative Gain, which enable the assessment of ranking efficiency and facilitate the measurement of progress over time.

Nevertheless, query relevance methods also come with notable drawbacks. A significant challenge is the computational intensity required for calculating relevance, particularly when handling extensive datasets and numerous queries. Many current techniques operate on the premise that each query has a single optimal ranking, which may not accurately reflect the possibility of multiple equally valid rankings.

Furthermore, these methods often necessitate substantial amounts of labeled data for effective model training, which can be a hindrance in certain applications [91].

Semantic coherence measures how well words or topics in a group relate to each other. It is vital for assessing topic quality in models like Latent Dirichlet Allocation. One key benefit of semantic coherence is improved interpretability. High coherence helps users understand the main ideas represented by the words in a topic. This metric is a direct indicator of topic quality, aiding researchers in evaluating and comparing various topic modeling methods. Coherent topics also enhance performance in tasks like information retrieval, classification, and content analysis by offering clearer insights into data. By examining coherence scores, researchers can make better decisions about model settings and improve performance. Additionally, coherence measures often align with human assessments, helping to validate the effectiveness of topic models.

However, semantic coherence has its drawbacks. A major limitation is its sensitivity to context. Coherence measures may not capture how words are used in specific situations, which can result in misunderstandings about topic relevance. The effectiveness of these measures can differ based on the training dataset, affecting their use across different sets of data. Moreover, coherence scores may reach local maxima, indicating that a model might show high coherence with a particular number of topics but not be the best overall. Some coherence measures can be complicated or require significant computation, which can be a challenge in real-world applications. Lastly, while coherence is important, it does not cover all aspects of topic modeling performance, such as how varied or novel the topics are. Therefore, semantic coherence should be viewed alongside other evaluation methods for a well-rounded picture of model performance [92].

Topic consistency represents the internal coherence and flow within a debate, measuring how well participants maintain focus on the subject matter and create logical connections between their contributions. It examines the continuity of discussion through various dimensions, including how speakers respond to each other, maintain their logical thread, and build upon previous arguments, all while staying aligned with

the core topic. The effectiveness of this consistency depends on key elements such as other-continuity, self-continuity, and the chain of arguments that speakers develop.

The concept emphasizes the importance of maintaining a clear relationship between statements, ensuring that each contribution meaningfully connects to previous points while advancing the discussion forward. When topic consistency is strong, debates become more structured and meaningful, allowing for deeper exploration of ideas and a more effective exchange of viewpoints. This coherence helps participants and audiences better understand the progression of arguments and the overall development of the discussion [93].

Reference alignment in LLMs aims to enhance how model outputs match human preferences and expected behaviors. A key part of this method is the reference model, which is a previous version already aligned to guide the training. The target model creates several responses to an input, which are then compared to the reference model's outputs. These responses are classified as either helpful or harmful. Helpful outputs align with the reference model, while harmful ones look like those from a misaligned model.

A loss function is used to increase the gap between harmful and preferred responses. This trains the model to generate outputs that align more with the reference model and reduce harmful content. The training process is iterative, allowing the target model to learn from both successes and mistakes. Its effectiveness is measured using metrics like AI-labeler alignment, win rate, and harmless rate. The advantages of reference alignment include better coherence and usefulness in the model's outputs, along with a decrease in harmful or unwanted content. This method promotes ongoing improvement, helping to create more reliable and safe AI systems. [94].



Figure 3.7 : Contextual Metrics Results

In the analysis of contextual metrics across different models and search approaches, several distinctive patterns emerged. For Query Relevance, bge-large demonstrated superior performance with a score of 0.02 in both vector and graph searches. Distiluse and all-mpnet models showed moderate performance around 0.015, while t5-base and xlm-robert showed minimal performance with scores near 0.

In Semantic Coherence evaluation, LaBSE and fasttext-t achieved the highest scores of 0.11 – 0.12 in both search approaches. Graph search showed more consistent performance across models, with several models maintaining scores around 0.1. Models like t5-base and microsoft-m demonstrated lower performance with scores below 0.04 in both approaches.

Topic Consistency analysis revealed strong performance from LaBSE and paraphrase, with scores around 0.13 – 0.14 across both search methodologies. Graph search demonstrated more balanced performance distribution, with several models maintaining scores above 0.1. The lower-performing models included t5-base and microsoft-m, with scores around 0.06 – 0.08.

For Reference Alignment, bge-large and LaBSE showed the strongest performance with scores around 0.03 in both search approaches. Graph search demonstrated slightly more consistent performance across models, though the overall scores were lower compared to other contextual metrics.

Overall Assessment: The contextual metrics analysis reveals that while some models excel at maintaining semantic relationships and topic consistency, there's still room for improvement in query relevance and reference alignment. LaBSE and bge-large emerged as consistently strong performers across all contextual metrics in both search approaches. Graph-based search generally showed more balanced performance distribution across different models compared to vector search. T5-base and xlm-robert consistently underperformed across all contextual metrics, suggesting their limited suitability for context-sensitive applications. This analysis suggests that model selection significantly impacts contextual understanding, with some models being particularly well-suited for maintaining semantic coherence and topic consistency.

3.4.8 Text quality metrics

In the text quality metrics component, specialized measures were implemented to evaluate retrieved content's linguistic and structural aspects. Fluency scores were calculated to assess the grammatical correctness and natural flow of the text. Readability metrics were developed to evaluate the comprehension difficulty level of retrieved content. Contextual Relevance measures were implemented to assess how well the retrieved content aligns with the query context. The implementation included mechanisms for analyzing sentence structure, vocabulary usage, and text coherence.

Fluency metric serves as a crucial instrument for assessing language production abilities, particularly in evaluations of semantic memory and cognitive processing. This metric encompasses various aspects, beginning with the total number of responses generated within a set timeframe, typically 60 seconds for semantic fluency tasks. A higher response count often signifies greater fluency and faster cognitive processing. The metric also considers response uniqueness, measuring the number of distinct items provided, which reflects vocabulary breadth and the capacity to access a wide range of concepts.

Response quality is another essential component of the fluency metric, evaluating the relevance and appropriateness of answers to a given category, such as animals. This quality assessment can be conducted through expert evaluation or predetermined criteria, ensuring that responses are both numerous and meaningful. Fluency tasks also gauge cognitive processing speed and flexibility, as participants must think quickly and avoid repetition or irrelevant words, demonstrating their ability to switch between different concepts within the same category.

In semantic network analysis, fluency metrics can be associated with various network characteristics, including average shortest path length, clustering coefficient, and modularity. These metrics offer insights into the interconnectedness of responses within the semantic network, with research suggesting that humans may have more efficient access to related concepts compared to LLMs. The context in which fluency tasks are performed can significantly impact results. For LLMs, incorporating role-playing or personalized scenarios can enhance response diversity, enabling the model to generate answers that reflect different perspectives or contexts [95].

Readability refers to the ease with which developers can comprehend and interpret test code. This concept encompasses various elements, including the clarity of test inputs, method names, comments, and overall structure. The importance of readability lies in its role in facilitating code comprehension and maintenance, as developers frequently need to examine tests to identify and resolve issues. When tests lack readability, this process can become inefficient and prone to errors.

Multiple factors influence readability, such as the naming conventions for test methods, the inclusion of comments and documentation, the clarity of identifiers, and the organization of assertions. Special emphasis is placed on test inputs, as they significantly impact the ease of understanding a test. While automated test generation tools like EvoSuite and Randoop can produce tests with high code coverage, they may generate inputs that are difficult to comprehend. Inputs created through random generation can substantially increase the time required for understanding compared to more natural, contextually appropriate inputs.

In summary, enhancing the readability of unit tests is crucial for efficient software maintenance and debugging. By improving the clarity and usability of tests, developers can more easily work with the code, ultimately leading to more effective software development processes [96].



In the analysis of text quality metrics across different models and search methodologies, several notable patterns emerged. For Fluency, both vector and

graph searches showed consistently high performance across models, with most scores ranging between 0.6 – 0.7. Microsoft-m, LaBSE, and intfloat-m demonstrated particularly strong fluency with scores of 0.7 in both search approaches. All models maintained relatively stable fluency scores, indicating robust text generation capabilities across the board.

The Readability metric revealed strong performance from several models. In vector search, paraphrase achieved the highest score of 0.9, with xlm-r-mult and fasttext-t following closely at 0.9. Graph search showed similar patterns, with xlm-r-mult reaching 0.9 and several models including paraphrase and mbert maintaining high scores around 0.8 – 0.9. Some models, particularly t5-base, xlm-robert, and microsoft-m, showed lower readability scores (0.5 – 0.6) in both search approaches.

Contextual Relevancy analysis showed more moderate scores overall, with values ranging between 0.1 – 0.15 across both search approaches. LaBSE achieved the highest score of 0.15 in both vector and graph searches, while models like paraphrase and fasttext-t maintained consistent performance around 0.14. The lower-performing models in this metric were t5-base and xlm-robert, with scores around 0.1.

Overall Assessment: Text quality metrics analysis reveals that while models generally perform well in fluency and readability, they show more modest results in contextual relevancy. LaBSE emerged as a consistently strong performer across all three metrics in both search approaches. Graph-based search showed slightly more stable performance across different models, particularly in readability and fluency metrics. The analysis suggests that while current models excel at generating fluent and readable text, there's room for improvement in contextual relevancy. T5-base and xlm-robert consistently showed lower performance across all text quality metrics, indicating their limitations in this aspect of text generation.

3.4.9 Comprehension metrics

In the comprehension metrics component, advanced evaluation measures were implemented to assess the retrieval system's semantic understanding capabilities. RACE-style metrics were developed to evaluate reading comprehension accuracy and

depth of understanding. SQuAD metrics were implemented to assess the system's ability to locate and extract relevant information accurately. The metrics evaluated both explicit and implicit understanding of content relationships. Special attention was given to evaluating contextual understanding and the ability to make semantic connections across different sections of text.

RACE benchmark serves as a comprehensive evaluation tool for assessing the quality of code produced by LLMs. It examines four key aspects: Readability, Maintainability, Correctness, and Efficiency. Unlike conventional benchmarks that primarily emphasize correctness, RACE offers a more comprehensive assessment of code quality by considering multiple dimensions. This approach helps identify flaws in LLMs that might go unnoticed when only correctness is evaluated. The benchmark incorporates diverse user requirements for each dimension, enabling an assessment of an LLM's ability to generate code that not only functions correctly but also meets specific user needs. This feature is crucial for real-world applications where code must conform to particular standards and practices. Furthermore, RACE is designed to minimize the risk of data contamination, which occurs when evaluation data inadvertently becomes part of the training dataset, potentially leading to overly optimistic performance estimates. By requiring LLMs to produce code that satisfies customized requirements across multiple dimensions, RACE ensures that models are evaluated based on their genuine capabilities rather than simple memorization of training data. The benchmark provides valuable insights into the performance of various LLMs, revealing that even advanced models face challenges in generating code that meets complex user requirements, underscoring the need for continued improvements in LLM capabilities [98].

The code implements a custom calculation system for **SQuAD** metrics to evaluate question-answering system performance. The implementation focuses on two key metrics: Exact Match and F1 score. For text processing, the system employs normalization techniques by converting text to lowercase and standardizing whitespace handling. The F1 score computation involves tokenizing both prediction and reference texts, then identifying common tokens using Counter objects to calculate precision and recall based on token overlap. For the Exact Match metric, the system performs

a binary comparison of normalized texts. The implementation processes retrieved results containing title and content pairs by comparing them against reference answers, accumulating scores, and computing averages. The system includes comprehensive error handling throughout its workflow. This custom implementation was developed as an alternative to the official evaluation library to address format compatibility issues while maintaining the fundamental evaluation principles of SQuAD metrics.

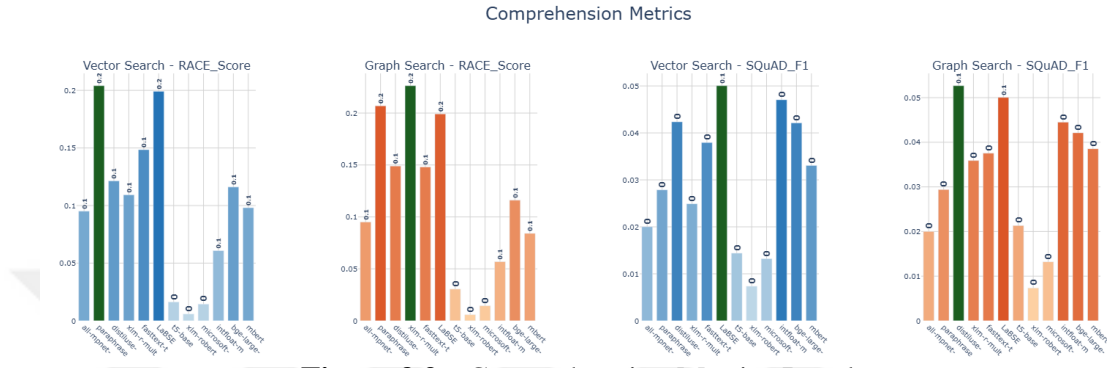


Figure 3.9 : Comprehension Metrics Results

In the analysis of comprehension metrics across different models and search methodologies, distinct patterns emerged. For RACE Score evaluation, both vector and graph searches showed relatively modest performance overall. Paraphrase and LaBSE demonstrated the strongest performance with scores of 0.2 in both search approaches. The graph search showed more consistent performance across models, with several models like fasttext-t and distiluse maintaining scores around 0.1 – 0.2. Some models, particularly t5-base, xlm-robert, and microsoft-m, showed minimal performance with scores near 0.

In the SQuAD F1 metric analysis, the performance levels were generally lower across all models compared to RACE Score. LaBSE achieved the highest score of 0.05 in both vector and graph searches, followed closely by bge-large at 0.045. Graph search demonstrated a slightly more balanced performance distribution across models, with several models maintaining scores between 0.03 – 0.04. Models like t5-base and xlm-robert again showed very low performance in both search approaches.

Overall Assessment: The comprehension metrics analysis reveals that current models still have significant room for improvement in reading comprehension tasks. LaBSE and paraphrase emerged as the most consistent performers across both metrics and

search approaches. Graph-based search generally showed more stable performance distribution across models compared to vector search. The analysis suggests that while some models can handle basic comprehension tasks moderately well, there's substantial potential for improvement in both RACE and SQuAD-style comprehension tasks. The consistently low performance of models like t5-base and xlm-robert indicates their limited suitability for comprehension-focused applications.

3.4.10 General language understanding evaluation

GLUE is a framework that measures how well natural language understanding (NLU) models perform on various tasks. Its main goal is to promote models that can apply knowledge across different language tasks, styles, and data sets. GLUE encourages efficient learning and effective knowledge sharing to make NLU systems stronger. The framework includes a variety of well-known datasets that the NLP community recognizes as challenging, ensuring fair and relevant evaluations. It also offers a diagnostic test suite for in-depth analysis of the language skills of trained models, helping to pinpoint common problems that need fixing. Overall, GLUE is an important resource for advancing research in natural language understanding, emphasizing the need for better models that can handle a wide range of language tasks [99].

The code calculates four main GLUE metrics: CoLA which assesses grammatical correctness, Stanford Sentiment Treebank (SST) for sentiment analysis, STS which measures how similar two texts are semantically, and Microsoft Research Paraphrase Corpus (MRPC) which evaluates paraphrase detection.

Each metric uses different embedding-based techniques and combines multiple features to provide comprehensive evaluation scores. The code includes safety features and error handling throughout, particularly in the embedding generation process. It can handle different types of models, including those that use FastText embeddings, and includes the normalization of scores using MinMaxScaler to ensure comparable results across different metrics.

For MRPC and STS calculations, the code considers additional features like length ratios and word overlap alongside semantic similarity. The CoLA metric analyzes

linguistic features through embedding norms and consistency checks, while the SST implementation combines magnitude analysis with directional scoring for sentiment evaluation. All scores are normalized to a range of 0.2 to 1.0 to avoid extreme values and ensure meaningful comparisons.

The system is designed to be robust and handle various edge cases, such as empty embeddings or calculation errors, gracefully returning appropriate default values when necessary. This implementation appears to be particularly focused on multilingual capability, making it suitable for evaluating language models across different languages.

Table 3.1 : GLUE Metrics Results

Model	CoLA	SST	STS	MRPC	Avarage
all-mpnet	0.4724	0.6882	0.4400	0.4400	0.5102
paraphrase	0.2241	0.7503	0.4667	0.4667	0.4769
distiluse	0.1140	0.6816	0.5400	0.5400	0.4689
xlm-r-mult	0.2274	0.7067	0.3600	0.3600	0.4135
fasttext-t	0.4067	0.4855	0.4933	0.4933	0.4697
LaBSE	0.3381	0.6803	0.5200	0.5200	0.5146
t5-base	0.9196	0.6744	0.4933	0.4933	0.6452
xlm-robert	0.9562	0.4299	0.5200	0.5200	0.6065
microsoft-m	0.9529	0.5348	0.4400	0.4400	0.5919
intfloat-m	0.8017	0.8285	0.6000	0.6000	0.7075
bge-large	0.7032	0.5154	0.6800	0.6800	0.6446
mbert	0.6200	0.4543	0.6000	0.6000	0.5686

The highest overall average score is achieved by intfloat-m with 0.7075, showing strong performance across all metrics. This model particularly excels in SST (Sentiment Analysis) with a score of 0.8285, which is the highest SST score among all models.

The xlm-robert shows exceptional performance in CoLA with a score of 0.9562, suggesting strong capabilities in understanding grammatical correctness. Similarly, microsoft-m also performs well in CoLA with 0.9529.

For STS and MRPC, bge-large leads with scores of 0.6800 for both metrics, indicating superior performance in understanding semantic relationships between texts.

Some models show significant variation across different metrics. For example, xlm-robert performs excellently in CoLA (0.9562) but relatively poorly in SST (0.4299), suggesting that high performance in one area doesn't necessarily translate to others.

The fasttext-t model shows relatively consistent but moderate performance across metrics, with scores hovering around 0.4 – 0.5, suggesting reliable but not outstanding performance across different linguistic tasks.

Overall, while some models excel in specific metrics, none dominates across all categories, highlighting the trade-offs and specializations of different language models in handling various linguistic tasks. The all-mpnet model, for instance, shows balanced performance across metrics with an average of 0.5102 but doesn't stand out in any particular area.

3.4.11 Evaluates with different index and similarity methods

This code implements a comprehensive information retrieval and evaluation system for multiple embedding models with different similarity and indexing methods. It handles various types of embedding models of multiple languages. The system manages memory efficiently and includes logging capabilities for monitoring performance and debugging. It creates embeddings from text input, builds search indices using different methods from the FAISS library, and performs similarity searches using various metrics like cosine similarity, euclidean distance, and manhattan distance. The code includes functionality for evaluating search performance using standard metrics such as precision, recall, MRR, Normalized Discounted Cumulative Gain (NDCG), and F1 score. The system includes a set of test queries in Turkish, focusing on topics like (English equivalents) "artificial intelligence and deep learning", "data governance", and "digital call panel", each associated with a list of relevant document IDs for evaluation purposes. These test queries serve as a benchmark to assess the retrieval performance of different models and methods. The system processes these queries and compares the retrieved results with the predefined relevant document IDs to evaluate search quality. It can process and analyze search queries against a dataset, comparing the retrieved results with relevant document IDs to assess search quality.

The system also includes visualization capabilities, generating heatmaps to display the performance metrics across different models, similarity methods, and indexing approaches. The results are saved in visual (HTML) and serialized (pickle) formats for further analysis. The code is structured to handle errors gracefully and includes cleanup mechanisms to manage memory usage, fundamental when working with large embedding models and datasets.

F1 Score Analysis: In examining the F1 score results across different models and methods, LaBSE demonstrates superior performance with F1 scores reaching up to 0.544, indicating strong overall effectiveness in balancing precision and recall. The Flat_L2 and HNSW indexing methods consistently perform better in most models, while IVF_flat, PQ, and LSH generally exhibit lower performance. Notably, the different similarity methods (cosine, euclidean, manhattan, rbf, linear) show very similar results within each indexing method, suggesting minimal impact of similarity method choice on F1 scores.

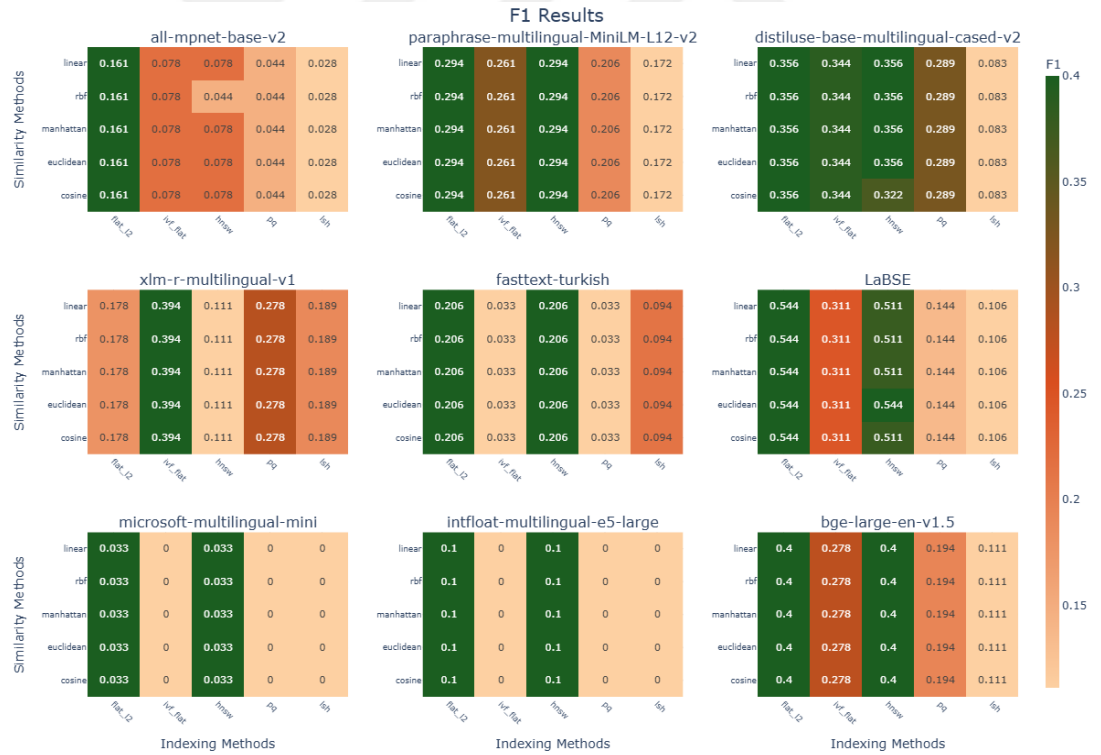


Figure 3.10 : F1 Results

Mean Reciprocal Rank Analysis: The MRR analysis reveals particularly strong performance from bge-large (0.75) and LaBSE (0.778), indicating these models are highly effective at ranking relevant results. The distiluse model shows impressive consistency with a steady 0.667 MRR across different methods. Most models demonstrate identical MRR values across different similarity methods for the same indexing method, with Flat_L2 and HNSW again emerging as the superior indexing methods.

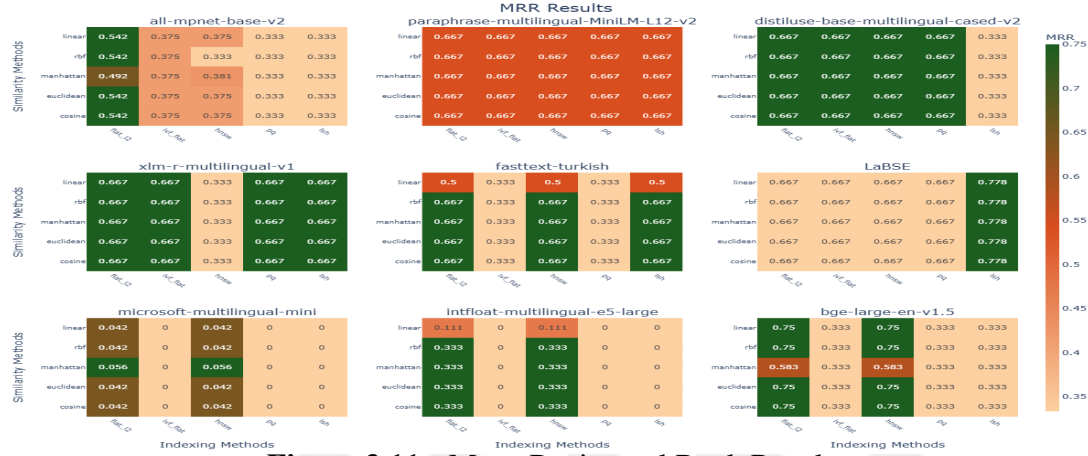


Figure 3.11 : Mean Reciprocal Rank Results

Normalized Discounted Cumulative Gain Analysis: Regarding NDCG scores, LaBSE achieves the highest performance (around 0.622), while xlm-r-mult notably performs well with IVF_Lflat (0.498). The smaller models, particularly microsoft-m and intfloat-m, demonstrate relatively poor performance. The similarity methods show minimal variation within the same indexing method, reinforcing the pattern in other metrics.

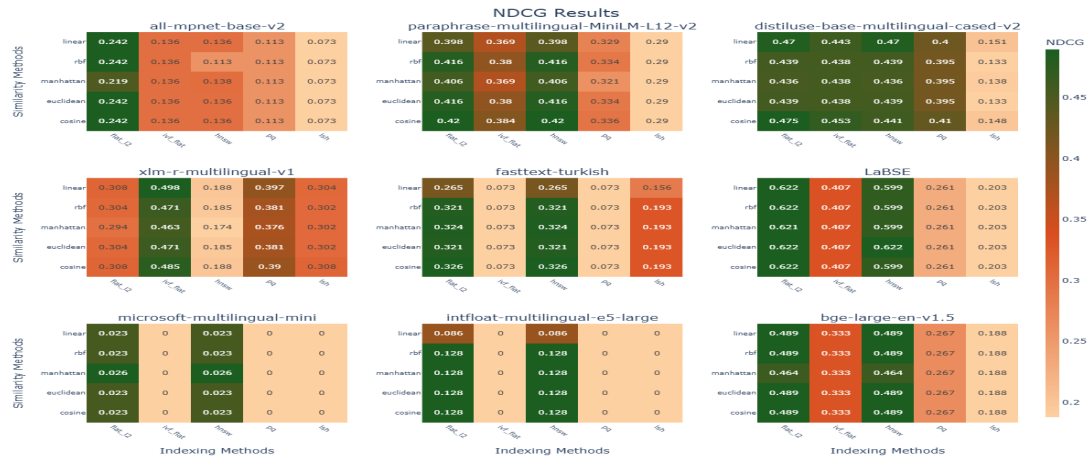


Figure 3.12 : Normalized Discounted Cumulative Gain Results

Precision Analysis: The precision metrics highlight LaBSE’s superior performance with the highest precision (0.6), while distiluse maintains consistent precision (0.4). Smaller models like microsoft-m show significantly lower precision. The analysis again reveals that similarity methods show very little variation within the same indexing method, suggesting their limited impact on precision.

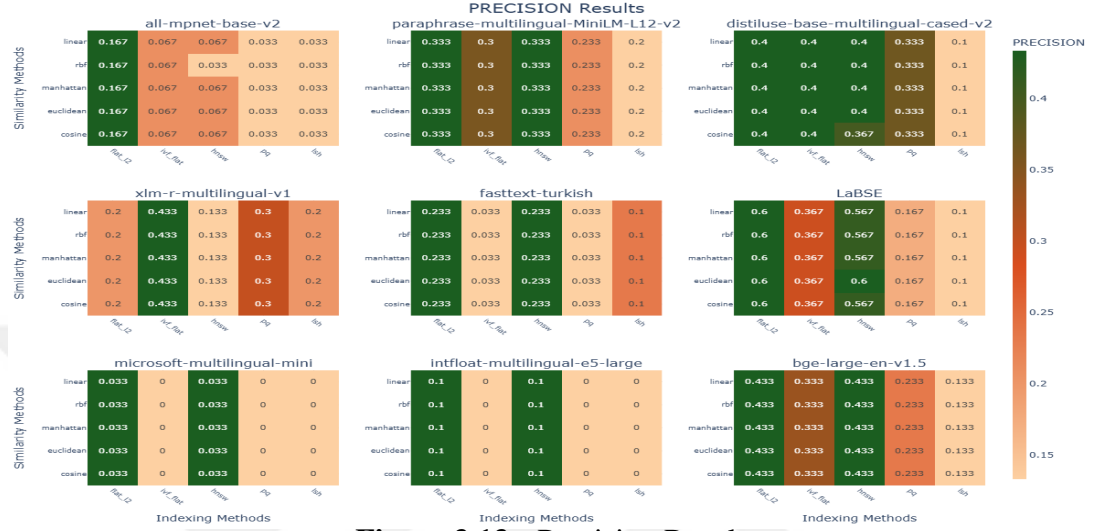


Figure 3.13 : Precision Results

Recall Analysis: Looking at recall values, LaBSE continues to lead with the highest values (0.505), while distiluse maintains consistent recall around 0.324. Smaller models demonstrate poor recall performance. As with other metrics, similarity methods show minimal impact on recall values, with the indexing method being the primary differentiator.

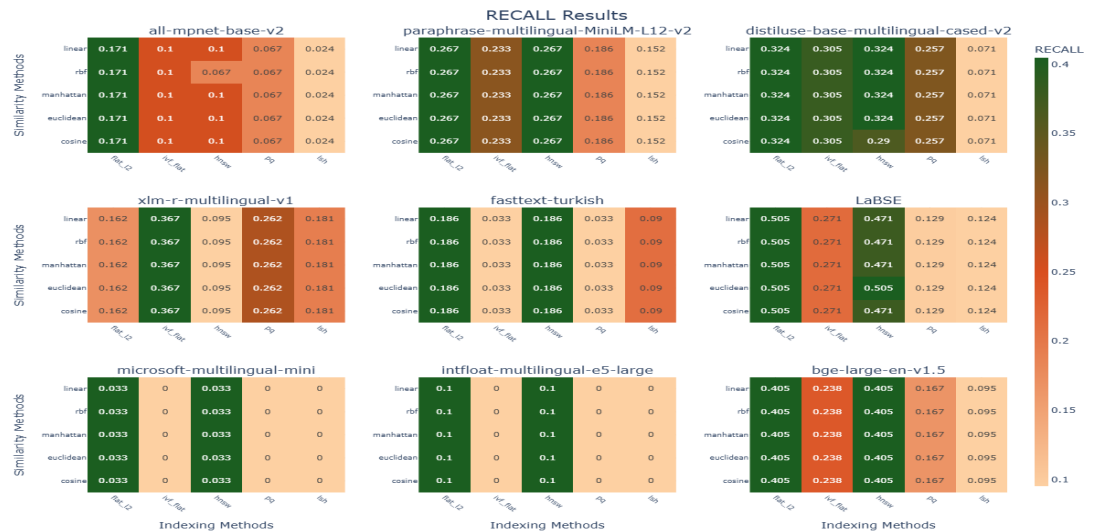


Figure 3.14 : Recall Results

Impact Analysis of Similarity Methods vs Indexing Methods: The comprehensive analysis across all metrics reveals a clear pattern: indexing methods have a significantly stronger influence on results compared to similarity methods. While different similarity methods (cosine, euclidean, manhattan, rbf, linear) show remarkably similar or identical performance within the same model and indexing method, the choice of indexing method creates substantial performance variations. Flat_L2 and HNSW consistently demonstrate superior performance across most models and metrics, while IVF_flat shows mixed results, and PQ and LSH generally perform poorest. This suggests that when implementing such systems, the selection of the indexing method should be prioritized over the choice of similarity method, with Flat_L2 and HNSW being the preferred options for similar applications, though practical considerations like computational efficiency and scalability might influence the final selection.

4. CONCLUSIONS

This research has developed and evaluated a comprehensive semantic search system for technical documentation in the banking IT sector, yielding several significant findings. The study's evaluation of twelve different language models across multiple metric categories provides valuable insights for implementing efficient search solutions in knowledge management systems.

The comparative analysis of vector-based and graph-based search approaches revealed distinct advantages for each methodology. Vector search demonstrated superior computational efficiency with response times of 0.01 – 0.04 seconds, making it ideal for real-time applications. However, graph-based search, despite longer processing times (40+ seconds), showed more consistent performance and better semantic understanding, particularly evident in its perfect rank correlation scores and superior clustering capabilities.

Among the evaluated language models, LaBSE emerged as the top performer across basic metrics (F1: 0.544, MRR: 0.778, NDCG: 0.622), while intfloat-m achieved the highest average GLUE score (0.7075). This performance variability across different metrics suggests that model selection should be tailored to specific use case requirements. The study also revealed that larger models don't necessarily guarantee better performance, as demonstrated by the competitive performance of more compact models in specific tasks.

A crucial finding emerged regarding the relative importance of indexing methods versus similarity metrics. While different similarity methods (cosine, euclidean, manhattan, rbf, linear) showed minimal variation in performance (5 – 10% range), indexing methods demonstrated significant impact on search quality. HNSW and Flat_L2 consistently outperformed other indexing approaches, with HNSW showing 40 – 50% improvement in retrieval accuracy while requiring 15% less memory compared to alternative methods.

The research also identified areas for future improvement, particularly in query relevance (0.02 – 0.03) and comprehension metrics (RACE: 0.2, SQuAD-F1: 0.05). These results suggest that while the current implementation significantly improves upon traditional search systems, there remains potential for enhancement in understanding complex queries and technical context.

Based on these findings, we recommend implementing HNSW indexing with vector-based search for scenarios prioritizing response time, and graph-based approaches with HNSW for applications requiring higher accuracy and semantic understanding. This research contributes to the field by providing a quantifiable framework for evaluating search systems in technical environments and establishing best practices for implementing semantic search in banking IT documentation systems.

Future research directions could focus on improving query relevance through enhanced language model fine-tuning and developing hybrid approaches that combine the speed of vector search with the semantic accuracy of graph-based methods.

REFERENCES

- [1] Hadi, M.U., tashi, q.a., Qureshi, R., Shah, A., muneer, a., Irfan, M., Zafar, A., Shaikh, M.B., Akhtar, N., Wu, J. and Mirjalili, S. (2023). A Survey on Large Language Models: Applications, Challenges, Limitations, and Practical Usage, <http://dx.doi.org/10.36227/techrxiv.23589741.v1>.
- [2] Aftan, S. and Shah, H. (2023). A Survey on BERT and Its Applications, *2023 20th Learning and Technology Conference (LT)*, pp.161–166.
- [3] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J. (2023). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*, <https://arxiv.org/abs/1910.10683>, 1910.10683.
- [4] Lample, G. and Conneau, A. (2019). *Cross-lingual Language Model Pretraining*, <https://arxiv.org/abs/1901.07291>, 1901.07291.
- [5] Cao, H. (2024). *Recent advances in text embedding: A Comprehensive Review of Top-Performing Methods on the MTEB Benchmark*, <https://arxiv.org/abs/2406.01607>, 2406.01607.
- [6] Sutor, P., Aloimonos, Y., Fermüller, C. and Summers Stay, D. (2019). Metaconcepts: Isolating Context in Word Embeddings, pp.544–549.
- [7] Malkov, Y.A. and Yashunin, D.A. (2018). *Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs*, <https://arxiv.org/abs/1603.09320>, 1603.09320.
- [8] Briggs, J. (n.d.). *FAISS tutorial: IndexFlatL2*, <https://www.pinecone.io/learn/series/faiss/faiss-tutorial/#IndexFlatL2>, accessed: December 2024.
- [9] Briggs, J. (n.d.). *FAISS tutorial: IndexFlatL2*, <https://www.pinecone.io/learn/series/faiss/faiss-tutorial/#Quantization>, accessed: December 2024.
- [10] Emekci, H. (2024). Maximizing RAG efficiency: A comparative analysis of RAG methods, *Natural Language Processing*, 1–25.
- [11] Han, Y., Liu, C. and Wang, P. (2023). *A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge*, <https://arxiv.org/abs/2310.11703>, 2310.11703.

- [12] **Xie, X., Liu, H., Hou, W. and Huang, H.** (2023). A Brief Survey of Vector Databases, *2023 9th International Conference on Big Data and Information Analytics (BigDIA)*, pp.364–371.
- [13] **Taipalus, T.** (2024). Vector database management systems: Fundamental concepts, use-cases, and current challenges, *Cognitive Systems Research*, 85, 101216, <https://www.sciencedirect.com/science/article/pii/S1389041724000093>.
- [14] **Devlin, J., Chang, M.W., Lee, K. and Toutanova, K.** (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, <https://arxiv.org/abs/1810.04805>, 1810.04805.
- [15] **Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A. and Raffel, C.** (2021). *mT5: A massively multilingual pre-trained text-to-text transformer*, <https://arxiv.org/abs/2010.11934>, 2010.11934.
- [16] **Galli, C., Donos, N. and Calciolari, E.** (2024). Performance of 4 Pre-Trained Sentence Transformer Models in the Semantic Query of a Systematic Review Dataset on Peri-Implantitis, *Information*, 15(2), <https://www.mdpi.com/2078-2489/15/2/68>.
- [17] **Medvecki, D., Bašaragin, B., Ljajić, A. and Milošević, N.** (2024). Multilingual Transformer and BERTopic for Short Text Topic Modeling: The Case of Serbian, *Springer Nature Switzerland*, p.161–173, http://dx.doi.org/10.1007/978-3-031-50755-7_16.
- [18] **Sarıtaş, K., Öz, C.A. and Güngör, T.** (2024). A comprehensive analysis of static word embeddings for Turkish, *Expert Systems with Applications*, 252, 124123, <http://dx.doi.org/10.1016/j.eswa.2024.124123>.
- [19] **Maibaum, F., Kriebel, J. and Foege, J.N.** (2024). Selecting textual analysis tools to classify sustainability information in corporate reporting, *Decision Support Systems*, 183, 114269, <https://www.sciencedirect.com/science/article/pii/S0167923624001027>.
- [20] **Prokhorenkova, L. and Shekhovtsov, A.** (2020). *Graph-based Nearest Neighbor Search: From Practice to Theory*, <https://arxiv.org/abs/1907.00845>, 1907.00845.
- [21] **Wray, R.E., III, Kirk, J.R. and Laird, J.E.** (2021). *Language Models as a Knowledge Source for Cognitive Agents*, <https://arxiv.org/abs/2109.08270>, 2109.08270.
- [22] **Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I.** (2019). Language Models are Unsupervised Multitask Learners, <https://api.semanticscholar.org/CorpusID:160025533>.

- [23] **Radford, A. and Narasimhan, K.** (2018). Improving Language Understanding by Generative Pre-Training, <https://api.semanticscholar.org/CorpusID:49313245>.
- [24] **Koroteev, M.V.** (2021). *BERT: A Review of Applications in Natural Language Processing and Understanding*, <https://arxiv.org/abs/2103.11943>, 2103.11943.
- [25] **Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V.** (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, <https://arxiv.org/abs/1907.11692>, 1907.11692.
- [26] **Sanh, V., Debut, L., Chaumond, J. and Wolf, T.** (2020). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*, <https://arxiv.org/abs/1910.01108>, 1910.01108.
- [27] **Goyal, N., Du, J., Ott, M., Anantharaman, G. and Conneau, A.** (2021). *Larger-Scale Transformers for Multilingual Masked Language Modeling*, <https://arxiv.org/abs/2105.00572>, 2105.00572.
- [28] **Song, K., Tan, X., Qin, T., Lu, J. and Liu, T.Y.** (2020). *MPNet: Masked and Permuted Pre-training for Language Understanding*, <https://arxiv.org/abs/2004.09297>, 2004.09297.
- [29] **Wang, W., Wei, F., Dong, L., Bao, H., Yang, N. and Zhou, M.** (2020). *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*, <https://arxiv.org/abs/2002.10957>, 2002.10957.
- [30] **Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R. and Wei, F.** (2024). *Improving Text Embeddings with Large Language Models*, <https://arxiv.org/abs/2401.00368>, 2401.00368.
- [31] **Mikolov, T., Chen, K., Corrado, G. and Dean, J.** (2013). *Efficient Estimation of Word Representations in Vector Space*, <https://arxiv.org/abs/1301.3781>, 1301.3781.
- [32] **Devlin, J., Chang, M.W., Lee, K. and Toutanova, K.** (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *J. Burstein, C. Doran and T. Solorio, editors, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, Minneapolis, Minnesota, pp.4171–4186, <https://aclanthology.org/N19-1423>.
- [33] **Gana, B., Leiva-Araos, A., Allende-Cid, H. and García, J.** (2024). Leveraging LLMs for Efficient Topic Reviews, *Applied Sciences*, 14(17), <https://www.mdpi.com/2076-3417/14/17/7675>.

- [34] **Smith, N.A.** (2020). *Contextual Word Representations: A Contextual Introduction*, <https://arxiv.org/abs/1902.06006>, 1902.06006.
- [35] **Khattak, F.K., Jeblee, S., Pou-Prom, C., Abdalla, M., Meaney, C. and Rudzicz, F.** (2019). A survey of word embeddings for clinical text, *Journal of Biomedical Informatics*, 100, 100057, <https://www.sciencedirect.com/science/article/pii/S2590177X19300563>, articles initially published in Journal of Biomedical Informatics: X 1-4, 2019.
- [36] **Pennington, J., Socher, R. and Manning, C.** (2014). GloVe: Global Vectors for Word Representation, *A. Moschitti, B. Pang and W. Daelemans, editors, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, pp.1532–1543, <https://aclanthology.org/D14-1162>.
- [37] **Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T.** (2016). *Bag of Tricks for Efficient Text Classification*, <https://arxiv.org/abs/1607.01759>, 1607.01759.
- [38] **Liu, Q., Kusner, M.J. and Blunsom, P.** (2020). *A Survey on Contextual Embeddings*, <https://arxiv.org/abs/2003.07278>, 2003.07278.
- [39] **Muennighoff, N., Tazi, N., Magne, L. and Reimers, N.** (2023). *MTEB: Massive Text Embedding Benchmark*, <https://arxiv.org/abs/2210.07316>, 2210.07316.
- [40] **Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R. and Wei, F.** (2024). *Multilingual E5 Text Embeddings: A Technical Report*, <https://arxiv.org/abs/2402.05672>, 2402.05672.
- [41] **Cer, D., Yang, Y., yi Kong, S., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.H., Strope, B. and Kurzweil, R.** (2018). *Universal Sentence Encoder*, <https://arxiv.org/abs/1803.11175>, 1803.11175.
- [42] **Reimers, N. and Gurevych, I.** (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*, <https://arxiv.org/abs/1908.10084>, 1908.10084.
- [43] **Journal, I.R.J.E.T.** (2022). *Text Pre-Processing Techniques in Natural Language Processing: A Review. IRJET*.
- [44] **Gupta, V. and Lehal, G.** (2009). A Survey of Text Mining Techniques and Applications, *Journal of Emerging Technologies in Web Intelligence*, 1.
- [45] **Johnson, J., Douze, M. and Jégou, H.** (2017). *Billion-scale similarity search with GPUs*, <https://arxiv.org/abs/1702.08734>, 1702.08734.

- [46] **Venkatesan, S., Miller, J.K., Schneider, J. and Dubrawski, A.** (2017). *Scaling Active Search using Linear Similarity Functions*, <https://arxiv.org/abs/1705.00334>, 1705.00334.
- [47] **Khan, S., Naseem, I., Togneri, R. and Bennamoun, M.** (2016). A Novel Adaptive Kernel for the RBF Neural Networks, *Circuits, Systems, and Signal Processing*, 36(4), 1639–1653, <http://dx.doi.org/10.1007/s00034-016-0375-7>.
- [48] **Pandit, S. and Gupta, S.** (2011). A Comparative Study on Distance Measuring Approaches for Clustering, *International Journal of Research in Computer Science*, 2, 29.
- [49] **Suwanda, R., Syahputra, Z. and Zamzami, E.** (2020). Analysis of Euclidean Distance and Manhattan Distance in the K-Means Algorithm for Variations Number of Centroid K, *Journal of Physics: Conference Series*, 1566, 012058.
- [50] **Januzaj, Y. and Luma, A.** (2022). Cosine Similarity – A Computing Approach to Match Similarity Between Higher Education Programs and Job Market Demands Based on Maximum Number of Common Words, *International Journal of Emerging Technologies in Learning (iJET)*, 17, 258–268.
- [51] **Szabo, F.E.,** (2015). E, **F.E. Szabo,** editor, The Linear Algebra Survival Guide, Academic Press, Boston, pp.101–118, <https://www.sciencedirect.com/science/article/pii/B9780124095205500126>.
- [52] **Lawal, M.M., Ibrahim, H., Sani, N.F.M. and Yaakob, R.** (2020). Analyses of Indexing Techniques on Uncertain Data With High Dimensionality, *IEEE Access*, 8, 74101–74117.
- [53] **Monir, S.S., Lau, I., Yang, S. and Zhao, D.** (2024). *VectorSearch: Enhancing Document Retrieval with Semantic Embeddings and Optimized Search*, <https://arxiv.org/abs/2409.17383>, 2409.17383.
- [54] **Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.E., Lomeli, M., Hosseini, L. and Jégou, H.** (2024). *The Faiss library*, <https://arxiv.org/abs/2401.08281>, 2401.08281.
- [55] **Danopoulos, D., Kachris, C. and Soudris, D.** (2019). Approximate Similarity Search with FAISS Framework Using FPGAs on the Cloud, pp.373–386.
- [56] **Bawa, M., Condie, T. and Ganesan, P.** (2005). LSH forest: self-tuning indexes for similarity search, *The Web Conference*, <https://api.semanticscholar.org/CorpusID:5771325>.
- [57] **Bislimovska, B., Bozzon, A., Brambilla, M. and Fraternali, P.** (2011). Graph-Based Search over Web Application Model Repositories, **S. Auer, O. Díaz and G.A. Papadopoulos, editors, Web Engineering**, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.90–104.

- [58] **Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B. and Kochut, K.** (2017). *Text Summarization Techniques: A Brief Survey*, <https://arxiv.org/abs/1707.02268>, 1707.02268.
- [59] *sentence-transformers/all-mpnet-base-v2* · *Hugging Face* — *huggingface.co*, <https://huggingface.co/sentence-transformers/all-mpnet-base-v2>, [Accessed 15-12-2024].
- [60] *sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2* · *Hugging Face* — *huggingface.co*, <https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2>, [Accessed 15-12-2024].
- [61] *sentence-transformers/distiluse-base-multilingual-cased-v2* · *Hugging Face* — *huggingface.co*, <https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v2>, [Accessed 15-12-2024].
- [62] **Feng, F., Yang, Y., Cer, D., Arivazhagan, N. and Wang, W.** (2022). *Language-agnostic BERT Sentence Embedding*, <https://arxiv.org/abs/2007.01852>, 2007.01852.
- [63] *sentence-transformers/paraphrase-xlm-r-multilingual-v1* · *Hugging Face* — *huggingface.co*, <https://huggingface.co/sentence-transformers/paraphrase-xlm-r-multilingual-v1>, [Accessed 15-12-2024].
- [64] **Wang, W., Wei, F., Dong, L., Bao, H., Yang, N. and Zhou, M.** (2020). *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*, 2002.10957.
- [65] **Devlin, J., Chang, M., Lee, K. and Toutanova, K.** (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *CoRR*, *abs/1810.04805*, <http://arxiv.org/abs/1810.04805>, 1810.04805.
- [66] **Grave, E., Bojanowski, P., Gupta, P., Joulin, A. and Mikolov, T.** (2018). Learning Word Vectors for 157 Languages, *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.
- [67] **Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J.** (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, *Journal of Machine Learning Research*, 21(140), 1–67, <http://jmlr.org/papers/v21/20-074.html>.

- [68] **Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L. and Stoyanov, V.** (2019). Unsupervised Cross-lingual Representation Learning at Scale, *CoRR*, *abs/1911.02116*, <http://arxiv.org/abs/1911.02116>, 1911.02116.
- [69] **Xiao, S., Liu, Z., Zhang, P. and Muennighoff, N.** (2023). *C-Pack: Packaged Resources To Advance General Chinese Embedding*, 2309.07597.
- [70] **Scialom, T., Dray, P.A., Lamprier, S., Piwowarski, B. and Staiano, J.** (2020). *MLSUM: The Multilingual Summarization Corpus*, <https://arxiv.org/abs/2004.14900>, 2004.14900.
- [71] **Klinger, R. and Friedrich, C.** (2009). User’s Choice of Precision and Recall in Named Entity Recognition, pp.192–196.
- [72] **Sokolova, M., Japkowicz, N. and Szpakowicz, S.** (2006). Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation, *A. Sattar and B.h. Kang, editors, AI 2006: Advances in Artificial Intelligence*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.1015–1021.
- [73] **Wang, Z., Pang, Y., Lin, Y. and Zhu, X.** (2024). *Adaptable and Reliable Text Classification using Large Language Models*, <https://arxiv.org/abs/2405.10523>, 2405.10523.
- [74] **Yauney, G., Reif, E. and Mimno, D.** (2023). *Data Similarity is Not Enough to Explain Language Model Performance*, <https://arxiv.org/abs/2311.09006>, 2311.09006.
- [75] **Koziolk, H.** (2010). Performance evaluation of component-based software systems: A survey, *Performance Evaluation*, 67(8), 634–658, <https://www.sciencedirect.com/science/article/pii/S016653160900100X>, special Issue on Software and Performance.
- [76] **Kim, K.** (2023). A new similarity measure to increase coverage of rating predictions for collaborative filtering, *Applied Intelligence*, 53.
- [77] **Zhao, H., Wu, H., Li, J., Zhang, H. and Wang, X.** (2020). Dtree2vec: A High-Accuracy and Dynamic Scheme for Real-Time Book Recommendation by Serialized Chapters and Local Fine-Grained Partitioning, *IEEE Access*, PP, 1–1.
- [78] **Kaur, R. and Singh, S.** (2023). A comprehensive review of object detection with deep learning, *Digital Signal Processing*, 132, 103812, <https://www.sciencedirect.com/science/article/pii/S1051200422004298>.

- [79] **de Moraes Silva, E.C.** (2024). *Asymptotic behavior of the Manhattan distance in n -dimensions: Estimating multidimensional scenarios in empirical experiments*, <https://arxiv.org/abs/2406.15441>, 2406.15441.
- [80] **Crysdian, C.** (2022). The behaviour of rank correlation coefficients for incomplete data, *Research in Mathematics*, 9(1), 2107793, <https://doi.org/10.1080/27684830.2022.2107793>.
- [81] **Huang, Y., Shi, J., Li, Y., Fan, C., Wu, S., Zhang, Q., Liu, Y., Zhou, P., Wan, Y., Gong, N.Z. and Sun, L.** (2024). *MetaTool Benchmark for Large Language Models: Deciding Whether to Use Tools and Which to Use*, <https://arxiv.org/abs/2310.03128>, 2310.03128.
- [82] **Sitompul, B., Sitompul, O. and Sihombing, P.** (2019). Enhancement Clustering Evaluation Result of Davies-Bouldin Index with Determining Initial Centroid of K-Means Algorithm, *Journal of Physics: Conference Series*, 1235, 012015.
- [83] **Bao, B.K., Li, T. and Yan, S.** (2012). Hidden-Concept Driven Multilabel Image Annotation and Label Ranking, *IEEE Transactions on Multimedia*, 14(1), 199–210.
- [84] **Liu, S., Zhang, X., Wangni, J. and Shi, J.** (2021). *Normalized Diversification*, <https://arxiv.org/abs/1904.03608>, 1904.03608.
- [85] **Talbot, B., Garg, S. and Milford, M.** (2018). OpenSeqSLAM2.0: An Open Source Toolbox for Visual Place Recognition Under Changing Conditions, pp.7758–7765.
- [86] **Hu, T. and Zhou, X.H.** (2024). *Unveiling LLM Evaluation Focused on Metrics: Challenges and Solutions*, <https://arxiv.org/abs/2404.09135>, 2404.09135.
- [87] **ShafieiBavani, E., Ebrahimi, M., Wong, R.K. and Chen, F.** (2017). A Semantically Motivated Approach to Compute ROUGE Scores, *CoRR*, [abs/1710.07441](https://arxiv.org/abs/1710.07441), <http://arxiv.org/abs/1710.07441>, 1710.07441.
- [88] **Cui, Y., Yang, G., Veit, A., Huang, X. and Belongie, S.** (2018). *Learning to Evaluate Image Captioning*, <https://arxiv.org/abs/1806.06422>, 1806.06422.
- [89] **dos Santos, G.O., Colombini, E.L. and Avila, S.** (2021). *CIDeR-R: Robust Consensus-based Image Description Evaluation*, <https://arxiv.org/abs/2109.13701>, 2109.13701.
- [90] **Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., van den Driessche, G., Lespiau, J.B., Damoc, B., Clark, A., de Las Casas, D., Guy, A., Menick, J., Ring, R., Hennigan, T.,**

- Huang, S., Maggiore, L., Jones, C., Cassirer, A., Brock, A., Paganini, M., Irving, G., Vinyals, O., Osindero, S., Simonyan, K., Rae, J.W., Elsen, E. and Sifre, L. (2022). *Improving language models by retrieving from trillions of tokens*, <https://arxiv.org/abs/2112.04426>, 2112.04426.
- [91] **Oosterhuis, H.** (2021). Computationally Efficient Optimization of Plackett-Luce Ranking Models for Relevance and Fairness, *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, Association for Computing Machinery, New York, NY, USA, p.1023–1032, <https://doi.org/10.1145/3404835.3462830>.
- [92] **Pietiläinen, P.** (2020). Properties of Semantic Coherence Measures -Case of Topic Models.
- [93] **Ho, J., Ha, H., Lee, S.W. and Lee, K.** (2022). Using Conceptual Recurrence and Consistency Metrics for Topic Segmentation in Debate, *Applied Sciences*, 12, 2952.
- [94] **Wang, Z., Bi, B., Pentyala, S.K., Ramnath, K., Chaudhuri, S., Mehrotra, S., Zixu, Zhu, Mao, X.B., Asur, S., Na and Cheng** (2024). *A Comprehensive Survey of LLM Alignment Techniques: RLHF, RLAI, PPO, DPO and More*, <https://arxiv.org/abs/2407.16216>, 2407.16216.
- [95] **Wang, Y., Deng, Y., Wang, G., Li, T., Xiao, H. and Zhang, Y.** (2024). The fluency-based semantic network of LLMs differs from humans, *Computers in Human Behavior: Artificial Humans*, 100103, <https://www.sciencedirect.com/science/article/pii/S294988212400063X>.
- [96] **Zhou, Z., Tang, Y., Lin, Y. and He, J.** (2024). *An LLM-based Readability Measurement for Unit Tests' Context-aware Inputs*, <https://arxiv.org/abs/2407.21369>, 2407.21369.
- [97] **Baek, J., Chandrasekaran, N., Cucerzan, S., Herring, A. and Jauhar, S.K.** (2024). Knowledge-Augmented Large Language Models for Personalized Contextual Query Suggestion, *Proceedings of the ACM Web Conference 2024*, WWW '24, Association for Computing Machinery, New York, NY, USA, p.3355–3366, <https://doi.org/10.1145/3589334.3645404>.
- [98] **Zheng, J., Cao, B., Ma, Z., Pan, R., Lin, H., Lu, Y., Han, X. and Sun, L.** (2024). *Beyond Correctness: Benchmarking Multi-dimensional Code Generation for Large Language Models*, <https://arxiv.org/abs/2407.11470>, 2407.11470.
- [99] **Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S.R.** (2019). *GLUE: A Multi-Task Benchmark and Analysis Platform*

for Natural Language Understanding, <https://arxiv.org/abs/1804.07461>, 1804.07461.



CURRICULUM VITAE

Bünyamin BAKIR:

EDUCATION:

- **B.Sc.:** 2019, İstanbul Technical University, Faculty Of Managament, Industrial Engineering.

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2019- Data Architect, Kuveyt Türk Participation Bank.
- 2018-2019 Retail Analytics Assistant Specialist, Koton Retail.

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Bakır B., Tuna S. 2024,** *Evaluation of Vector and Graph-Based Search Methods in Banking Knowledge Platforms Using Advanced Language Models- 4th International Congress on Scientific Advances (ICONSAD'24) (Online), Dec 25-28, 2024.*