CM2

Table of Contents

Quiz 1 Rubric

- CMQuiz1.pdf Read it

Computing for Medicine, Lecture 6: Word Vectors and Embeddings for Computing in Medicine

Course: Computing for Medicine
Google Classroom Code: dnd5qkt5
Semester: Monsoon 2025
Lecture: 6

International Classification of Diseases (ICD)

The International Classification of Diseases (ICD) is a globally used diagnostic classification system for epidemiology, health management, and clinical purposes.

ICD-10 Chapter Classifications

Code	ICD-10 Chapter	Range
I	Certain infectious and parasitic diseases	A00-B99
II	Neoplasms	C00-D48
III	Diseases of the blood and blood-forming organs and certain disorders involving the immune system	D50-D89
IV	Endocrine, nutritional and metabolic diseases	E00-E90
V	Mental and behavioral disorders	F00-F99
VI	Diseases of the nervous system	G00-G32
VII	Diseases of the eye and adnexia	H00-H59
VIII	Diseases of the ear and mastoid process	H60-H95

Code	ICD-10 Chapter	Range
IX	Diseases of the circulatory system	I00-I99
X	Diseases of the respiratory system	J00-J99
XI	Diseases of the digestive system	K00-K93
XII	Diseases of the skin and subcutaneous tissue	L00-L99
XIII	Diseases of the musculoskeletal system and connective tissue	M00-M99

ICD Coding for Respiratory System (J00-J99)

Section	Code Range
Acute upper respiratory infections	J00-J06
Influenza and pneumonia	J10-J18
Other acute lower respiratory infections	J20-J22
Other diseases of the upper respiratory tract	J30-J39
Chronic lower respiratory diseases	J40-J47
Lung diseases due to external agents	J60-J70
Other respiratory diseases principally affecting the interstitium	J80-J84
Suppurative and necrotic conditions of the lower respiratory tract	J85-J86
Other diseases of the pleura	J90-J94
Other diseases of the respiratory system	J95-J99

Basic Idea Behind All Modern Representations

Modern representation learning follows a pipeline:

```
Input Space → Feature Space → Abstract Representation Learning (Deep Learning) → Output Space
```

This process transforms raw data into meaningful representations that can be processed by machine learning algorithms.

Word Embeddings

Definition

Word embeddings are mathematical representations of language units that capture semantic meaning and relationships between words.

Types of Word Representations

1. **Basic vectorization approaches**
2. **Distributed representations**
3. **Universal language representation**
4. **Handcrafted features**

Motivation

The goal is to capture the meaning of language rather than just structure.

Process Steps

1. Break the sentence into lexical units such as lexemes, words, and phrases
2. Derive the meaning for each of the lexical units
3. Understand the syntactic (grammatical) structure of the sentence

4. Understand the context in which the sentence appears

Featurization Methods

One Hot Encoding

One Hot Encoding maps each word to a unique position in a vector where only one element is 1 and all others are 0.

Process

- Map each word w → a unique integer wid between 1 and |V|
- Each word becomes a V-dimensional binary vector
- Example: Dog = [1 0 0 0 0 0]
- "Dog Bites Man" = [[1 0 0 0 0 0] [0 1 0 0 0 0] [0 0 1 0 0 0]]

Implementation Example

```
sentences = ["It was the best of times",
"it was the worst of times",
"it was the age of wisdom",
"it was the age of foolishness"]

tokenized_sentences = [[t for t in sentence.split()] for sentence in sentences]
vocabulary = set([w for s in tokenized_sentences for w in s])

import pandas as pd
pd.DataFrame([[w, i] for i,w in enumerate(vocabulary)])
```

```
def onehot_encode(tokenized_sentence):
    return [1 if w in tokenized_sentence else 0 for w in vocabulary]

onehot = [onehot_encode(tokenized_sentence)
          for tokenized_sentence in tokenized_sentences]

for (sentence, oh) in zip(sentences, onehot):
    print("%s: %s" % (oh, sentence))
```

Output Example

```
[0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1]: It was the best of times
[1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0]: it was the worst of times
[0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0]: it was the age of wisdom
[0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0]: it was the age of foolishness
```

Challenges of One Hot Encoding

- Size of a one-hot vector ∝ size of the vocabulary |V|
- Sparse representations - matrix full of zeroes
- Storage constraints and overfitting due to sparsity
- Does not give fixed length representation
- Assumes independence between words
- Out of Vocabulary (OOV) problem - needs retraining every time a new word is added

Bag of Words (BoW)

Concept

- Assumes that text belonging to a given class in the dataset is characterized by a unique set of words

- Knowing the words present in a text tells about the class (bag)
- Each document is a V-dimensional vector
- Example: "Dog Bites Man" = [1 1 1 0 0 0]

Advantages

- Gives a fixed length representation
- Captures some semantic similarity of documents

Challenges with BoW Representation

- Sparsity
- Same word may mean different things in different contexts
- Out of Vocabulary (OOV) words
- Order information is lost

Bag of N-Grams (BoN)

Purpose

Attempts to preserve context and order information by considering sequences of words.

Example

- Bigrams = {dog bites, bites man, man bites, bites dog, dog eats, eats meat, man eats, eats food}
- "Dog Bites Man" = [1,1,0,0,0,0,0,0]

Types

- Bigram (2 words)
- Trigram (3 words)
- N-gram (n words)

Discriminative vs Generative AI

Discriminative Models

- Learn p(y|x) - probability of output given input
- Focus on decision boundaries

Generative Models

- Learn p(x,y) - joint probability distribution
- Can generate new data samples

Text Generation Models (Large Language Models)

Modern language models can take text input and generate coherent text output, such as summarizing articles.

Probabilistic Autoregressive Models (Traditional Models)

Concept

Traditional language models assign probabilities to word sequences.

Example

For the sentence "John bought a book":

- p(John, bought, a, book) = 0.02
- p(book, bought, a, John) = 0.01
- p(book, a, John, bought) = 0.0001

Sequential Generation

```
John bought a
John bought a book
John bought a book for
John bought a book for coloring
```

Mathematical Formulation

p(John, bought, a, book) = p(John) × p(bought | John) × p(a | John, bought) × p(book | John, bought, a)

Vector Space Paradigm: Distributional Hypothesis

Core Quote

"You shall know a word by the company it keeps!" - Firth (1957)

Distributional Representation

Inducing distributional properties from context to generate a representation. Words that appear in similar contexts tend to have similar meanings.

Core Idea: Vector Space

Key Principles

- Represent words (or tokens) as vectors
- Words with similar meanings have related vector representations
- Associations between words are captured in shared weights
- Vector weights can be trained using neural networks

Word Algebra

Vector representations allow mathematical operations on words:

```
KING - MAN + WOMAN = QUEEN
UNCLE - MAN + WOMAN = AUNT
```

This demonstrates that word vectors capture semantic relationships and analogies.

How Embeddings Capture Context

Word embeddings can distinguish between different meanings of the same word based on context. For example, the word "lie" has multiple meanings:

Different Contexts of "Lie"

Untruth (verb)

- "I will lie for personal benefit"
- "Rob reveals to Tracy that everything was a lie and that he still hated her"

Mathematical Sense (verb)

- "A skew polygon does not lie in a flat plane, but zigzags in three (or more) dimensions"
- "As an open string propagates through spacetime, its endpoints are required to lie on a D-brane"

Lie Down (verb)

- "There Fenrir will lie until Ragnarok"
- "They lie down to sleep deeply"

Geographical (island) - (verb)

- "Some 3,579 islands lie adjacent to the peninsula"
- "The islands lie on the Kerguelen Plateau in the Indian Ocean"

Conceptual Placement (verb)

- "According to Dewey, conversation, debate and dialogue lie at the heart of a democracy"
- "The origins of mathematical thought lie in the concepts of number, magnitude and form"

Geographical (other) - (verb)

- "Very small portions lie within the Pueblo County School District 70"
- "The ruins of the town lie along the river Ziz in the Tafilalt oasis near the town of Rissani"

Word2Vec (Generation 1)

Word2Vec is a popular method for creating word embeddings using neural networks.

Continuous Bag of Words (CBOW)

Concept

- Predicts the middle word given the context
- Assigns probability to sentences such that "good" sentences are maximized

Example

```
The quick brown fox jumped over the lazy dog
```

Training with CBOW

Source Text: "The quick brown fox jumps over the lazy dog."

Training Samples (context, target):

- ((quick, brown), The)
- ((The, brown, fox), quick)
- ((The, quick, fox, jumps), brown)
- ((quick, brown, jumps, over), fox)

Skip-Gram Variant

Concept

Predicts the context given the middle word.

Example

```
The quick brown fox jumped over the lazy dog
```

Training with Skip-Gram

Source Text: "The quick brown fox jumps over the lazy dog."

Training Samples (target, context):

- (the, quick)
- (the, brown)
- (quick, the)
- (quick, brown)
- (quick, fox)
- (brown, the)
- (brown, quick)
- (brown, fox)
- (brown, jumps)
- (fox, quick)
- (fox, brown)
- (fox, jumps)
- (fox, over)

Summary

Word vectors represent a fundamental shift from traditional symbolic representations to distributed representations that capture semantic meaning. The evolution from one-hot encoding to sophisticated embedding methods like Word2Vec demonstrates the progression toward more meaningful and efficient representations of language for computational processing.

Key takeaways:

1. Traditional methods like one-hot encoding and bag-of-words have limitations in capturing semantic relationships
2. Distributional hypothesis forms the foundation for modern word embeddings
3. Word2Vec methods (CBOW and Skip-Gram) learn dense vector representations that capture semantic and syntactic relationships
4. Context plays a crucial role in determining word meaning and embeddings capture this through distributional patterns

Computing for Medicine, Lecture 7: Embeddings + Transformers

Google Classroom Code: `dnd5qkt5` Monsoon 2025

Word Vectors

Training with CBOW (Continuous Bag of Words)

Source Text:

```
The quick brown fox jumps over the lazy dog.
```

Repeated for multiple training samples.

Training Samples (context, target):

- (quick, brown) → The
- (The, brown, fox) → quick
- (The, quick, fox, jumps) → brown
- (quick, brown, jumps, over) → fox

CBOW Model Architecture

- **Input Layer:** words from context
- **Hidden Layer:** sums up context vectors
- **Output Layer:** predicts the target word
- Dimensionality:
  - N-dim (hidden)
  - V-dim (output)

---

word2Vec: Skip Gram Variant

- **Skip Gram** predicts the context given the middle word.
- Example text:

```
The quick brown fox jumped over the lazy dog.
```

Training Samples (target, context):

- (the, quick)
- (the, brown)
- (quick, the)
- (quick, brown)
- (quick, fox)
- (brown, the)
- (brown, quick)
- (brown, fox)
- (brown, jumps)
- (fox, quick)
- (fox, brown)
- (fox, jumps)
- (fox, over)

Continuous SkipGram Model Architecture

- **Input Layer:** target word
- **Hidden Layer:** context features
- **Output Layer:** context prediction
- Dimensionality:
  - N-dim (hidden)
  - V-dim (output)

Key Papers in Word Embeddings

Efficient Estimation of Word Representations in Vector Space

- **Authors:** Tomas Mikolov, Greg Corrado, Kai Chen, Jeffrey Dean
- **Published by:** Google Inc., Mountain View, CA

Distributed Representations of Words and Phrases and their Compositionality

- **Authors:** Kai Chen, Jeffrey Dean, Tomas Mikolov, Ilya Sutskever, Greg Corrado
- **Published by:** Google Inc.

Open Source Code: Multiple Implementations

- **word2vec project:**
  - Efficient implementation for continuous bag-of-words and skip-gram architectures
  - Used for vector representations of words
- **Quick Start:**
  - Download code via SVN:

    ```
    svn checkout http://word2vec.googlecode.com/svn/trunk
    ```

- **License:** Apache License 2.0

Words in Vector Space (Example Embedding)

Word	Embedding (example)

Word	Embedding (example)
pandemic	[0.3, 0.8]
covid	[0.3, 0.7]
hospital	[0.4, 0.5]
football	[0.9, 0.1]

Summary of Embeddings

- **Embeddings capture distributional similarities between words**
- Enable efficient word algebra and analogies
- **Popular Models**:
  - Word2Vec: Pre-trained neural network-based embedding (Unit = Word)
  - GloVe: Pretrained word embedding from Stanford (Unit = Word)
  - FastText: Pretrained embedding from Facebook (Unit = Character)

Pretrained Embeddings for Clinical Data and Concepts

Available Models

Name	Model	Data/Concepts	Terms	Dimension
PubMed-w2v.bin	word2vec	PubMed	2.4M	200
PMC-w2v.bin	word2vec	PubMed Central	25M	200
PubMed-and-PMC-w2v.bin	word2vec	PubMed, PubMed Central	4.1M	200
wikipedia-pubmed-and-PMC-w2v	word2vec	PubMed, PubMed Central, Wikipedia	5.5M	200
drug word embeddings	word2vec	PubMed, DrugBank	553,195	420
AWE-CM [49]	word2vec	UMLS CUI (concepts)	265M	300
claims_codes_hs_300 [55]	word2vec	ICD-9 codes (concepts)	51,327	300
claims_cuis_hs 300 [55]	word2vec	UMLS CUI (concepts)	14,852	300
cui2vec [56]	word2vec/Glove	UMLS CUI (concepts)	108,477	500
concept embeddings [58]	AiTextML	MeSH ID (concepts)	26,103	100
word embeddings [58]	AlTextML	PubMed	513,196	100
ELMO (PubMed model) [11]	ELMO	PubMed	N/A	1024
BioBERT [15]	BERT	PubMed	N/A	768/1024
ClinicalBERT [16,17]	BERT	MIMIC III	N/A	768

Key Points:

- **ELMO** models use character information
- **BERT** models use sub-word information
- Can represent any concept

Evaluation Tasks for Word Embeddings

- **Evaluation metrics:** relatedness, similarity, cluster quality, conceptual similarity, clinical information extraction, disease prediction, mortality prediction, NER, drug name recognition, high-risk prediction, phenotype classification, term abstraction, etc.
- **Studies:** De Vise et al., Chie et al., Dubois et al., Wang et al., etc.
- **Corpora examples:**
  - PubMed, PMC
  - Mayo Clinic notes, OHSUMED
  - Google News

- MedHelp forum
- Medical claims
- UMLS, MeSH
- MIMIC-III
- Wikipedia

Sequence Models and Transformers

Sequence to Sequence Modeling — Encoder-Decoder Framework

Example

- **INPUT:** Je suis étudiant
- **OUTPUT:** I am a student

Structure

- **Encoder:** Processes input sequence
- **Decoder:** Produces output sequence

Steps

1. Embedding words from input
2. Apply encoder at each time step
3. Use decoder to predict output words step by step

Attention and Transformer Architecture

"Attention Is All You Need" (Vaswani et al., 2017)

- **Main authors:** Ashish Vaswani, Noam Shazeer, Niki Parmar, Llion Jones, Aidan Gomez, Jakob Uszkoreit, Łukasz Kaiser, Illia Polosukhin

Transformer Structure (Block Diagram)

```
Nx layers:
- Add & Norm
- Multi-Head Attention
- Feed Forward
- Softmax Output
Input: Embedded sequence + Positional Encoding
Output: Embedded sequence + Positional Encoding
Masked Multi-Head Attention in decoder
```

Input Embeddings (Sample Words)

- microwave, refrigerator, bulb, kitchen, light, charger, battery, sink, bathroom, etc.

Positional Encoding

- Adds position information to input embeddings.

Attention Mechanism

Principle

- Context Vector Calculation:
  - For each output token, compute weighted sum over encoder hidden states.
  - Formula:

```
C_i = Σ a_ij h_j
a_ij = exp(e_ij) / Σ_k exp(e_ik)
e_ij = alignment score for i-th decoder state vs j-th encoder state
```

Self Attention

- Each word attends to other words in the sequence to capture context.
- Calculation cost: $O(n^2)$ (quadratic in sequence length)

Query, Key, and Value Representation

- For each input, derive three vectors:
  - Query (q)
  - Key (k)
  - Value (v)
- Compute attention scores as dot products of queries and keys:

```
Attention(Q, K, V) = softmax( Q × K^T / sqrt(d_k) ) × V
```

Masked Attention

- Prevents decoder from attending to future positions
- Look-ahead masking used in generation tasks (next word prediction)

Resources

- Code and downloads:
  - http://evexdb.org/pmresources/ngrams/PubMed/
  - http://evexdb.org/pmresources/ngrams/PMC/
  - http://evexdb.org/pmresources/vec-space-models/wikipedia-pubmed-and-PMC-w2v.bin
  - https://github.com/chop-dbhi/drug_word_embeddings
  - Visual Guides:
    - https://jalammar.github.io/illustrated-transformer/
    - https://becominghuman.ai/attention-is-all-you-need-16bf481d8b5c
    - http://peterbloem.nl/blog/transformers

End of Lecture

Thanks for attending the class!

Computing for Medicine, Lecture 8: Transformer (Continued)

Course: Computing for Medicine
Google Classroom Code: dnd5qkt5
Semester: Monsoon 2025
Lecture: 8 - Transformer (Continued)

Recap: Transformer Architecture

Complete Transformer Structure

The Transformer consists of two main components:

1. **Encoder Stack** (left side)
   - N identical layers (Nx)

- Each layer contains:
  - Multi-Head Attention
  - Add & Norm
  - Feed Forward
  - Add & Norm

2. **Decoder Stack** (right side)
   - N identical layers (Nx)
   - Each layer contains:
     - Masked Multi-Head Attention
     - Add & Norm
     - Multi-Head Attention (encoder-decoder attention)
     - Add & Norm
     - Feed Forward
     - Add & Norm

Key Components Flow

```
Input → Input Embedding → Positional Encoding → Encoder Stack
Output (shifted right) → Output Embedding → Positional Encoding → Decoder Stack
Decoder Stack → Linear → Softmax → Output Probabilities
```

Scaled Dot Product Attention (Similarity with Context)

Mathematical Formula

```
Attention(Q, K, V) = softmax(QK^T / √dk) V
```

Components Explained

- **Q (Query):** "What am I asking about?" - the search intent
- **K (Key):** "Where should I look?" - addressing mechanism
- **V (Value):** "What information do I retrieve once I've found the right spots?" - actual content

Processing Steps

1. **Matrix Multiplication:** QK^T
2. **Scaling:** Divide by √dk (where dk is dimension of keys)
3. **Softmax:** Convert to probability distribution
4. **Value Retrieval:** Multiply with V matrix

Visual Example

Example sentence: "like this movie very much !"

The attention mechanism processes this through:

- MatMul operation
- Scale by √dk (where d=5 in example)
- Apply SoftMax
- Optional Mask
- Final MatMul with Values

Mechanics of Attention - Step by Step

Step 1: Generate Similarity Score

Like "n judges scoring m keys" - creates similarity matrix between queries and keys.

Step 2: Scale and Softmax

- **Scaling:** Prevents extremely large values that could cause vanishing gradients
- **Softmax:** Turns each row into a probability distribution
- Each row sums to 1.0

Step 3: Retrieve the Values (Output Embeddings)

Use the probability weights to get weighted sum of value vectors.

Analogies for Understanding Query, Key, Value

Library Analogy

Scenario: You're searching for books in a library

- **Key (K):** Book's index card used to locate it
- **Query (Q):** Your search question
- **Value (V):** The actual book content you take home once you've matched the index card

Practical Example

When processing "he bought books":

- "he": 30% attention weight
- "bought": 50% attention weight
- "books": 0.1% attention weight
- "a": 0.1% attention weight
- "boy": 0.1% attention weight

Masked Attention

Purpose

Prevents the decoder from "looking ahead" at future tokens during training.

Look-Ahead Mask

```
Raw attention weights → Apply mask → Masked attention
```

Matrix Representation

The mask matrix has:

- 1s for allowed connections
- 0s for blocked connections (future positions)

Example mask pattern:

```
1 0 0 0 0
1 1 0 0 0
1 1 1 0 0
1 1 1 1 0
1 1 1 1 1
```

Sequence Processing

- y1 can only attend to x1
- y2 can attend to x1, x2
- y3 can attend to x1, x2, x3
- And so on...

This ensures that position i can only attend to positions less than i.

Multi-Head Attention

Architecture

```
Input → Linear Transformations (Q, K, V) → h parallel attention heads → Concat → Linear → Output
```

Mathematical Process

1. **Linear Transformations:** Apply separate linear layers to create Q, K, V for each head
2. **Parallel Processing:** Run h attention heads in parallel
3. **Concatenation:** Combine outputs from all heads
4. **Final Linear:** Apply final linear transformation

Key Insight

Mathematical principle: One big linear function, then splitting allows each subspace to compose with the full original vector. Splitting first and then applying linear function restricts the possibilities.

Benefits

- Allows model to attend to information from different representation subspaces
- Each head can learn different types of relationships
- Provides robustness and richer representations

Positional Encodings

Problem Statement

No convolution, no recurrence: Transformer has no inherent sense of position, so we need to add positional information.

Mathematical Formulas

For position `pos` and dimension `i`:

```
PE(pos, 2i) = sin(pos / 10000^(2i/d_model))
PE(pos, 2i+1) = cos(pos / 10000^(2i/d_model))
```

Key Properties

- **Even dimensions:** Use sine function
- **Odd dimensions:** Use cosine function
- Creates unique encoding for each position
- Allows model to learn relative positions

Visual Pattern

The positional encoding creates wave-like patterns across dimensions:

- Different frequencies for different dimensions
- Enables the model to distinguish between positions
- Maintains consistent mathematical relationships

Feed Forward Layers

Mathematical Formula

```
FFN(x) = max(0, xW1 + b1)W2 + b2
```

Components

- **First Linear Layer:** xW1 + b1
- **ReLU Activation:** max(0, ...)
- **Second Linear Layer:** ...W2 + b2

Key Characteristics

- **Fully Connected Dense Layers:** For learning complex patterns
- **Individual Networks:** Each FFN is different for each layer (Nx)
- **Original Paper:** Nx = 8 (8 encoder layers, 8 decoder layers)

Purpose

- Processes the attention output
- Adds non-linearity to the model
- Allows for complex feature transformations

Dropout

Application Points

Quote from original paper:

> "We apply dropout to the output of each sub-layer before it is added to the sub-layer and normalized. In addition, we apply dropout to the sums of embeddings and positional encodings in both encoder and decoder stacks."

Specific Locations

1. **Sub-layer outputs:** Before Add & Norm operations
2. **Embedding layers:** After combining embeddings with positional encodings
3. **Both stacks:** Applied in encoder and decoder

Purpose

- **Regularization:** Prevents overfitting
- **Robustness:** Makes model less dependent on specific neurons
- **Generalization:** Improves performance on unseen data

Complete Architecture Summary

Encoder Stack (Nx layers)

Each encoder layer contains:

1. Multi-Head Attention
2. Add & Norm
3. Feed Forward Network
4. Add & Norm

Decoder Stack (Nx layers)

Each decoder layer contains:

1. Masked Multi-Head Attention
2. Add & Norm
3. Multi-Head Attention (with encoder)
4. Add & Norm
5. Feed Forward Network
6. Add & Norm

Input Processing

- **Encoder Input:** Input Embedding + Positional Encoding
- **Decoder Input:** Output Embedding (shifted right) + Positional Encoding

Output Generation

- **Final Processing:** Linear layer → Softmax → Output Probabilities
- **Training:** Outputs shifted right for teacher forcing
- **Inference:** Autoregressive generation

Key Technical Details

Dimensions and Scaling

- **dk:** Dimension of key vectors
- **Scaling Factor:** √dk prevents attention weights from becoming too large
- **d_model:** Model dimension used throughout the architecture

Attention Types

1. **Self-Attention (Encoder):** Each position attends to all positions in input
2. **Masked Self-Attention (Decoder):** Each position attends only to earlier positions
3. **Encoder-Decoder Attention:** Decoder attends to encoder representations

Training vs Inference

- **Training:** Uses teacher forcing with shifted outputs
- **Inference:** Generates tokens one by one autoregressively
- **Masking:** Ensures no information leakage from future tokens

---

References and Additional Resources

- Original reference: http://nlp.seas.harvard.edu/2018/04/03/attention.html
- Positional encoding details: https://timodenk.com/blog/linear-relationships-in-the-transformers-positional-encoding/
- Transformer visualization: http://peterbloem.nl/blog/transformers

---

Thank you for attending the class!

Lecture 9: BERT — Bidirectional Encoder Representations from Transformers

Recap: Sequence-to-Sequence Models

Transformer architectures enable automated translation and text generation via encoder and decoder components:

- **Transformer Encoder**: Processes input data sequence (e.g., "I love llamas")
- **Transformer Decoder**: Generates output sequence; uses previously generated tokens
- **Self-Attention**: Computes context for each token
- **Feedforward Neural Networks**: Refine representations within both encoder and decoder
- **Masked Self-Attention**: Decoder only attends to known inputs and already generated outputs

---

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Authors: Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (Google AI Language)

Key Concepts

- **Contextualized Word Embeddings**: BERT encoder learns representations from large text corpus.
- **Bidirectional Encoding**: Encodes tokens considering both left and right context simultaneously.
- Implements only encoder and not decoder side; designed for representation, not generation.

Input Structure

- Special tokens: `[CLS]` for classification, `[MASK]` for masked prediction tasks
- Input sequence: `[CLS] I love llamas`
- BERT encoder applies self-attention, passing through multiple stacked layers
- Output: Contextualized word embeddings

Pre-training Process

1. **Masked Language Modeling (MLM):**
   - Randomly mask words in input
   - Model predicts masked words using context
2. **Next Sentence Prediction (NSP):**
   - Model learns coherence between sentence pairs

Fine-tuning Process

- BERT is fine-tuned for specific downstream NLP tasks with labeled datasets (e.g., classification, named entity recognition, paraphrase identification).

BERT & the GLUE Benchmark

- **General Language Understanding Evaluation (GLUE):**
  - Multiple NLP tasks are evaluated using BERT embeddings and architecture
  - Example input: `[CLS] My dog is cute. [SEP] He likes playing. [SEP]`
  - Embeddings:
    - **Token Embeddings:** Learned by BERT
    - **Segment Embeddings:** Identify sentence segments
    - **Position Embeddings:** Encode position information for each token
  - Final prediction layer (classification, question answering, etc.)

GLUE Tasks Examples

- GLUE includes various tasks like sentence similarity, sentiment analysis, natural language inference, etc.
- Scores: Individual tasks aggregate for final GLUE score

BERT Training Steps for GLUE

1. **Pretrain on large unlabeled dataset (e.g., Wikipedia, Books Corpus)**
2. **Fine-tune for target tasks**
   - Each task: Supervised training with labeled dataset
   - Examples: Classification, Named Entity Recognition (NER), Paraphrase Identification

---

Specialized BERT Models for Biomedical Domain

BioBERT

Original Paper: "BioBERT: a pre-trained biomedical language representation model for biomedical text mining" by Jinhyuk Lee, Wonjin Yoon, etc.

BioBERT Workflow

- **Pre-training:** Using biomedical corpora (PubMed abstracts, PMC full-text articles)
- **Fine-tuning:** For specific biomedical tasks (NER, Relation Extraction, Question Answering)

Pre-training Corpora

Corpus	Number of Words	Domain
English Wikipedia	2.5B	General
Books Corpus	0.8B	General
PubMed Abstracts	4.5B	Biomedical
PMC Full-text Articles	13.5B	Biomedical

Corpus Combinations

Model	Corpus Combination
BERT (Devlin et al.)	Wiki + Books
BioBERT (+PubMed)	Wiki + Books + PubMed
BioBERT (+PMC)	Wiki + Books + PMC
BioBERT (+PubMed+PMC)	Wiki + Books + PubMed + PMC

BioBERT Downstream Tasks

- **Named Entity Recognition**: NCBI disease, BC2GM datasets
- **Relation Extraction**: EU-ADR, ChemProt (example: "Variants in GENES contribute to DISEASES susceptibility")
- **Question Answering**: BioASQ challenge, e.g., "What does mTOR stand for?" → "Mammalian Target Of Rapamycin"

Clinical Domain-Specific BERT Models

ClinicalBERT & Embeddings

- **ClinicalBERT**: Fine-tuned on clinical notes from MIMIC III dataset
- **Discharge Summary BERT**: Trained on hospital discharge summaries
- **Bio+Clinical BERT**: Merges biomedical and clinical corpora for pre-training
- **MedNLI, i2b2 (2006–2014)**: Used as evaluation tasks for clinical embeddings

Performance Snapshot

Model	MedNLI	i2b2 2006	i2b2 2010	i2b2 2012	i2b2 2014
BERT	77.6%	93.9	83.5	75.9	92.8
BioBERT	80.8%	94.8	86.5	78.9	93.0
ClinicalBERT	80.8%	91.5	86.4	78.5	92.6
Discharge Summary BERT	80.6%	91.9	86.4	78.4	92.8
Bio+Clinical BERT	82.7%	94.7	87.2	78.9	92.5
Bio+Discharge Summary BERT	82.7%	94.8	87.8	78.9	92.7

UMLS BERT — Clinical Domain Knowledge Augmentation

UmlsBERT introduces domain knowledge using UMLS (Unified Medical Language System) semantic groups:

- **Semantic Group Embeddings (SG)**: Custom embedding matrix added to BERT input to encode medical concepts
  - Example: "heart" → Anatomy
  - Each UMLS group represented by separate embedding vector
- Comparison between vanilla BERT and UmlsBERT for biomedical domain recognition, e.g., "lungs" → specific semantic group

---

PEGASUS — Pre-training for Abstractive Summarization

Authors: Jingqing Zhang, Yao Zhao, Mohammad Saleh, Peter J. Liu

Core Idea

- **Gap Sentence Generation (GSG)**: Masks complete sentences, model must generate masked sentences
- **Seq2Seq Transformer Architecture**: Encoder-decoder structure for summarizing long-form text

Performance

- Evaluated using ROUGE scores (rouge1-F, rouge2-F, rougel-F)
- Datasets: XSum, CNN/DailyMail, WikiHow, Reddit TIFU
- Performance scales with size of training data and number of examples

Example: BBC News Article

- PEGASUS abstracts information (e.g., "four Royal Navy frigates") from source content listing specific ship names
- Can generate concise summaries of lengthy text content

Embeddings in Biomedical NLP

Concept and Word Embeddings

- Various models available for biomedical and clinical data
- Embeddings support representation for any word, concept, or term
- Examples: Word2vec, GloVe, ELMO, BERT variants

Name	Model	Data/Concepts	Terms	Dim.
PubMed-w2v.bin	word2vec	PubMed	2.4M	200
PMC-w2v.bin	word2vec	PubMed Central	25M	200
PubMed-PMC-w2v.bin	word2vec	PubMed+PMC	4.1M	200
wiki-pubmed-PMC-w2v.bin	word2vec	PubMed+PMC+Wikipedia	5.5M	200
drug word embeddings	word2vec	PubMed+DrugBank	553195	420
AWE-CM	word2vec	UMLS CUI (concepts)	265M	300
claims_codes_hs_300	word2vec	ICD-9 codes	51,327	300
claims_cuis_hs_300	word2vec	UMLS CUI (concepts)	14,852	300
cui2vec	word2vec/Glove	UMLS CUI (concepts)	108,477	500
concept embeddings	AiTextML	MeSH ID (concepts)	26,103	100
word embeddings	AiTextML	PubMed	513,196	100
ELMO (PubMed model)	ELMO	PubMed	NA	1024
BioBERT	BERT	PubMed	NA	768/1024
ClinicalBERT	BERT	MIMIC III	NA	768

Evaluation Tasks

- Word and concept similarity
- Disease prediction
- Named entity recognition
- Relation extraction
- Mortality prediction
- Readmission risk prediction
- Clinical information extraction

References and Useful Links

- PEGASUS: https://ai.googleblog.com/2020/06/pegasus-state-of-art-model-for.html
- BioBERT Paper: doi: 10.1093/bioinformatics/btz682
- BioBERT Code/Data: https://github.com/dmis-lab/biobert
- Drug Word Embeddings: https://github.com/chop-dbhi/drug_word_embeddings
- Evaluation Corpora: http://evexdb.org/pmresources/

Summary

BERT and its specialized extensions (BioBERT, ClinicalBERT, UmlsBERT, PEGASUS) have transformed NLP and biomedical text mining by providing powerful contextual embeddings and pre-training techniques. Fine-tuning BERT-based models on specialized corpora enables state-of-the-art performance on domain-specific tasks such as named entity recognition, relation extraction, and clinical information extraction.

Lecture 10: Shock Modes

AI for Multimodal Healthcare Data

Why AI in Healthcare?

AI aims to improve healthcare by analyzing complex medical data, enabling early diagnosis, optimized treatment, and better patient outcomes. Healthcare environments generate a variety of data types, which need advanced methods for interpretation and action.

Types of Healthcare Data

- **Structured Data:** Numeric, coded records.
- **Semi-Structured Data:** Forms, partially organized records.
- **Unstructured Data:** Text, images, audio.
- **Multimodal Data:** Combines more than one type (e.g., images + text, physiological measurements + notes).

Examples of Data Types

- **Video Observations:** Joint examinations.
- **Audio & Linguistic Data:** Patient interactions and interviews.
- **Imaging Data:** Neuroimaging, X-rays.
- **Physiological Measurements:** Cardiovascular assessments, blood pressure.
- **Patient Records:** Medical history, notes.
- **Genetic & Biomarker Data:** Lab tests, genetic screenings.
- **Behavioral & Lifestyle Data:** Physical activity, habits.
- **Wearables & Sensor Data:** Sleep analytics, fitness trackers, continuous monitoring devices.
- **Electronic Health Data:** Medication records, prescriptions.
- **Emergency & Response Data:** Records of emergency care and interventions.

Case Study – ShockModes

Shock: What is it?

Shock is a life-threatening condition in which body tissues are deprived of enough oxygen and nutrients. If left untreated, shock leads to organ failure.

Causes of Shock

- Severe loss of blood or fluids.
- Heart failure (unable to pump effectively).
- Infections causing blood vessel dilation and leakage.

Importance of Early Treatment

- Mortality rates for shock: **30 to 40%**.

Shock Index (SI)

- Formula: **SI = Heart Rate (HR) / Systolic Blood Pressure (SBP)**.

- Abnormal threshold: **SI ≥ 0.7** indicates higher risk of circulatory collapse.

ShockModes Goal

- Build an early warning system.
- Predict abnormal SI 24 hours in advance.
- Use both vitals and physician notes.

Workflow

1. **Data Collection:** Vitals and clinical notes obtained from the MIMIC-III database (large ICU-stay dataset).
2. **Data Cleaning & Preprocessing:** Remove noise, inconsistencies, and prepare data for feature extraction.
3. **Feature Extraction:**
    - **Vitals:** Analyze time-series patterns.
    - **Notes:** Extract keywords, generate embeddings (representations).
    - **Feature Fusion:** Combine numerical, textual, and categorical features.
4. **Cohort Building:** Group patient records into meaningful cohorts for analysis.
5. **Labeling:** Assign abnormal/normal SI labels.
6. **Supervised Learning:** Apply machine learning algorithms to learn from labeled data.
7. **Model Training, Testing, Evaluation, and Interpretability:**
    - Train models on prepared data.
    - Test performance and refine parameters.
    - Evaluate and interpret the results.

Dataset

- Used **MIMIC-III** database (17,294 ICU-stays).
- Cohorts combine clinical notes and vital signs for 24-hour windows.
- Key extracted vitals: Heart Rate (HR), Systolic Blood Pressure (SBP), Respiratory Rate (RR), SpO2.

MIMIC-III Clinical Database Details

- Hosted by PhysioNet, curated by Alistair Johnson, Tom Pollard, Roger Mark.
- Published: September 4, 2016 (Version 1.4).
- Data also available on Google Cloud Platform (GCP) and Amazon Web Services (AWS).
- Tutorials provided for cloud access and querying.

Features from Notes

- Clinical information extracted using standard healthcare terminologies (like SNOMED CT).
- Categories include: body structure, clinical findings, environment/location, events, observable entities, organisms, products (drugs/biologics), procedures, qualifiers, record artifacts, situations.
- SNOMED CT Browser used to standardize concepts for machine learning.
- Example: Clinical finding itemized by SCTID codes and described using domain taxonomy.

Features from Vitals

- **3117 time-series features** extracted using tsfresh (Python library).
- Examples of extracted features:
    - **Autocorrelation:** Detects repeating patterns in signals.
    - **Fourier Transform:** Identifies frequency content and changes in the signal.
    - **Entropy:** Quantifies randomness in signals.
    - **Mean, Variance, Coefficients:** Large suite of numeric descriptors of each vital sign time-series.
- Extracted features used in ML tasks: classification, regression, clustering, forecasting.

Model Development and Validation

- **Input Features:** Vitals (from tsfresh), embeddings of therapeutics and HOPI (History of Present Illness).
- **Feature Reduction:** Extra Trees classifier (from scikit-learn) selects most important features.
- **Imbalanced Data Handling:** SMOTE oversampling (from scikit-learn).
- **Models Used:** Logistic Regression, Random Forest, GradientBoost, AdaBoost, XGBoost.
- **Evaluation:** Bootstrap sampling over 100 iterations; key metrics are AUC-ROC and F1-score.

SHAP Analysis for Interpretability

- **SHAP (Shapley Additive Explanations):** Assigns importance values to each input feature, explaining their impact on predictions.
- **Interpretability:** Compares base (prior) probabilities from training data with predictions to explain outcomes.
- **Local Explanations:** Visualized by waterfall plots which show the effect of features on SI prediction for each patient.
- **Global Explanations:** Feature importance is visualized for the overall population using bar plots and beeswarm plots.

Example Feature List with SHAP Impact (selected features):

```
riss
heparin_sodium_prophylaxis
Chief Complaint
Assessment and Plan
Past medical history
vancomycin
ppi
metoprolol
plavix
acetaminophen
neuro
ns
oxycodone
02
propofol
keppra
lisinopril
lasix
insulin
...plus sum of 181 other features

Signal Processing Features:
final_abp_sys_fft_coefficient_attr_"abs"_coeff_37
final_abp_sys_fft_coefficient_attr_"angle"_coeff_24
resp_has_duplicate_min
spo2_fft_coefficient_attr_"real"_coeff_85
final_abp_sys_fft_coefficient_attr_"abs"_coeff_40
heparin_sodium_prophylaxis
...

SHAP value (impact on model output)
-1.5 ... 0.0 ... +1.0 (Low to High impact)
```

Features like medications, signal-processing outcomes (Fourier, autocorrelation, entropy values), and clinical history have direct impact on the model's predictions.

Evaluation Parameters

- **Precision:** TP/(TP+FP)
- **Recall:** TP/(TP+FN)
- **AUC-ROC Curve:** Area under the ROC curve to assess predictive ability.
- **F1-score:** Harmonic mean of precision and recall.

Summary

The ShockModes project utilized advanced multimodal AI methods, extracting thousands of features from both time-series vital signs and clinical notes. It built a system to predict circulatory collapse (abnormal SI) 24 hours ahead, leveraging state-of-the-art feature engineering, machine learning, and explainable AI to enable actionable predictions in critical care.

Automating the Transformation of Free-Text Clinical Problems into SNOMED CT Expressions

Authors

- Kevin J. Peterson, MS
    - Division of Information Management and Analytics, Mayo Clinic, Rochester, MN

    - Bioinformatics and Computational Biology Program, University of Minnesota, Minneapolis, MN
- Hongfang Liu, PhD
    - Department of Health Sciences Research, Mayo Clinic, Rochester, MN

Abstract

An important function of the patient record is to effectively and concisely communicate patient problems. In many cases, these problems are represented as short textual summaries and appear in sections like problem lists, diagnoses, and chief complaints. Free-text problem descriptions capture the clinician's intent well but are challenging for downstream analytics. This study presents an automated approach to convert free-text problems into structured Systematized Nomenclature of Medicine – Clinical Terms (**SNOMED CT**) expressions using advances in deep learning for semantic representation. The methods outperform current techniques in relation identification and highlight real-world challenges in clinical text processing.

Introduction

The shift to data-driven healthcare creates challenges in managing complex patient data. Before electronic health records (**EHRs**), there were struggles to standardize patient data. Dr. Weed's 1960s proposal for "problem lists" centralized patient conditions, shaping interaction with records and simplifying clinical narratives. These concise summaries also appear in diagnoses, chief complaints, and reasons for visit, providing an efficient way to express patient conditions.

Free-text is common for problem summaries, expressing clinical states directly. While free-text helps clinicians, its lack of structure hinders analytics. In contrast, controlled terminology brings structure but limits expressiveness. Many systems offer free-text entry even when codified capture is preferred, producing a tradeoff: free-text maximizes clinician usability, while structured forms enable analytics and data reuse.

This study proposes a framework for converting free-text problem descriptions to structured, codified formats using **Natural Language Processing (NLP)**. The structured representation, specifically **SNOMED CT Expressions**, improves downstream analytics and standardization.

Background & Related Work

Controlled Vocabulary in Clinical Problems

Using a controlled vocabulary for free-text clinical problems is an active research area.

- **SNOMED CT** is an effective standard for capturing these semantics.
- Two representation types:
    - **Pre-Coordinated Concept**: Single atomic unit/identifier.
        - Example: `370221004|Severe asthma (disorder)|`
    - **Post-Coordinated Concept**: Composition of multiple concepts, together conveying the semantics.
        - Example: `195967001|Asthma (disorder)| + 24484000|Severe (severity modifier)|`

Even summary-level problems are often too expressive for a single pre-coordinated concept. Studies show:

- SNOMED CT alone represented only 51.4% of problem entries without composition, but 92.3% with composition.
- 53% of summary-level data required two or more concepts.

Post-coordination uses SNOMED CT Compositional Grammar, which allows main concepts qualified by attribute/value pairs:

```
195967001|Asthma (disorder)|:
246112005|Severity (attribute)| = 24484000|Severe (severity modifier)|
```

Previous Approaches

- Focused on attribute relationship identification/classification (e.g., which attribute best describes "Severe" in "Severe asthma").
- One technique: iterative learning of lexical patterns for each relationship type.
- Another approach (Kate): relation identification at full phrase level, using Support Vector Machine (SVM) models.
- Contributions in this study:
    - End-to-end process, including all subtasks and NLP techniques like dependency parsing.
    - Leverage deep learning to improve relation identification versus SVM.
    - Initial evaluation using a large clinical corpus.

Methods

The process is split into three steps:

1. Concept Extraction

- Free-text problem is input.
- Use **MetaMap** (from NLM) for named-entity recognition to extract **UMLS** concepts.
- Example for "Venous varicosities in lower extremities with recent thrombophlebitis":

```
C0226813:Vein of lower extremity
C0042345:Varicosities
C0332185:Recent
C0040046:Thrombophlebitis
```

- Map UMLS concepts to SNOMED CT. Only SNOMED CT terms are retained.
- For concepts matching multiple SNOMED CT entries, all are considered.

2. Focus Concept Selection

- Select main semantic focal point from extracted SNOMED CT concepts.
- Use **dependency parsing** to align the root word of the problem description to a MetaMap concept.
- Tool: **spaCy** with biomedical models from **scispaCy**.
- Example: root word "varicosities" matched to relevant concept.

3. Relation Identification

- Identify relationships between the problem text and extracted concepts.
- Formal definition: Given a problem description and a concept, compute the suitable SNOMED CT attribute connecting them.
- Classifier outputs probabilities for SNOMED CT attribute types; incompatible ranges are pruned.
- Highest remaining probability is chosen.

Deep Learning Approaches

- **Bidirectional LSTM (BiLSTM)**: Enhanced RNN architecture processes info sequentially in both forward and reverse directions, enabling context inference.
- **Convolutional Neural Networks (CNN)**: Also processes spatial/text relationships, via sliding windows, used widely in images and NLP tasks.
- Both use **BERT** (Bidirectional Encoder Representations from Transformers), specifically "Clinical BERT" fine-tuned on clinical corpus.

Model Architecture

- Two inputs: full problem text and concept text.
- Inputs are embedded via BERT.
- Processed by BiLSTM (100 units) or CNN (two layers, 20% dropout).
- Fully connected dense layer with softmax outputs probabilities for attribute types.

Training

- Used SNOMED CT US Edition (Sept 2018) relationships for classifier training.
- Excluded "Is a" relationships; those are inferable from hierarchy.
- Training records for all pairs of source/target labels.
- Relationship types with fewer than 125 instances excluded.
- Final dataset: 1,526,043 records, 78 relationship types.
- Experiments used held-out test sets.

Evaluation

Measured performance for focus concept selection and relation identification.

- **Concept Extraction** not directly evaluated (MetaMap used). MetaMap performance reference: Reátegui et al. (2018).

Four experiments:

Experiment 1: BiLSTM & CNN vs Naïve Bayes

- Trained/tested on SNOMED CT Relationship data.
- 25% test, 75% train, 20% of train withheld for validation (deep learning models).
- F1 scores recorded for attributes and averages.
- Results:
  - Naïve Bayes: Accuracy 0.720, Macro F1 0.460, Weighted F1 0.665
  - CNN + Clinical BERT: Accuracy 0.886, Macro F1 0.822, Weighted F1 0.880
  - BiLSTM + Clinical BERT: Accuracy 0.888, Macro F1 0.851, Weighted F1 0.888

Experiment 2: BiLSTM vs Kate's SVM Model

- Used procedure from Kate (2013) to replicate for 5 attributes.
- For each, 5000 positive/negative examples.
- Recorded classifier accuracy for relation presence.
- BiLSTM generally outperformed SVM except "Has active ingredient" attribute.

Experiment 3: Real-World Text Corpus

- Used corpus of 14 million clinical documents from Mayo Clinic.
- 401 random problem descriptions annotated by three experts:
  - Focus concepts identified
  - Relationships (21 types) assigned between focus and modifiers
- Inter-annotator agreement (Cohen's kappa):
  - Focus: 0.78, 0.85, 0.84
  - Relationship: 0.76, 0.76, 0.82
- Tested on support relationships (>20 annotations). Results comparing Clinical Text F1 scores (real clinical text) and SNOMED CT Relationship F1 scores: | Attribute | Clinical Text F1 | SNOMED CT Relationship F1 | F1 Δ | |----------😐:----------😐-----😐 | Severity | 0.936 | 0.882 | 0.054| | Laterality | 0.970 | 0.999 | -0.029| | Clinical course | 0.889 | 0.994 | -0.105| | Finding site | 0.865 | 0.988 | -0.123| | Due to | 0.507 | 0.830 | -0.323| | Has interpretation | 0.638 | 0.992 | -0.354| | Following | 0.431 | 0.837 | -0.406| | Associated with | 0.218 | 0.738 | -0.520|

Experiment 4: Focus Concept Selection – Dependency Parsing Models

- Accuracy (agreement with annotators): | Model | Accuracy | |----------|----------| | Default spaCy English (baseline) | 0.68 | | ScispaCy Biomedical | 0.75 | | ScispaCy Biomedical + fine-tuning | 0.91 |

Discussion

Both CNN and BiLSTM outperformed the Naïve Bayes classifier for relationship identification. The BiLSTM slightly surpassed CNN in macro F1 average, which is crucial for weighting relationship types equally and avoiding class imbalance effects.

Deep learning approaches (BiLSTM) outperformed SVM in relation identification for most attributes. However, direct comparison is limited due to differences in attributes evaluated and exclusion of "Is a" relationships.

Performance on real clinical text was notably lower than on SNOMED CT-derived relationships, reflecting unpredictability and diversity in clinical entries versus structured terminology. Some attributes ("Due to", "Has interpretation", "Associated with") suffered greater accuracy drops (highly negative F1Δ), highlighting challenges in generalizing from controlled data to clinical scenarios.

Dependency parsing with domain-specific, fine-tuned models significantly improved focus concept selection.

Conclusion

This work developed an end-to-end system for converting unstructured clinical problem statements to SNOMED CT expressions, introducing a deep learning model for concept-relationship identification. This model outperformed existing approaches and highlighted the constraints of training on structured data alone for real clinical text processing.

Limitations

- No full gold-standard test set for text-to-SNOMED CT evaluation. Steps evaluated independently.

- Multiple valid syntactic representations for expressions.
- Physician coding exhibits considerable variation.
- These issues make quantitative evaluation difficult.

Acknowledgment

- Funded by grant NCATS U01TR02016.
- Thanks to Donna Ihrke, Luke Carlson, and Sunyang Fu for annotation and guideline development assistance.

References

For complete numbered references, see source document. (All in-document citations preserved exact order)

Language Models as Ontology Encoders

Authors & Affiliations

- Hui Yang (The University of Manchester)
- Jiaoyan Chen (The University of Manchester)
- Yuan He (Amazon, University of Oxford)
- Yongsheng Gao (SNOMED International)
- Ian Horrocks (University of Oxford)

Abstract

OWL ontologies formally represent complex knowledge and provide semantic reasoning in domains like healthcare and bioinformatics. Ontology embeddings infer plausible knowledge and approximate reasoning, but face key limitations:

- Geometric model-based embeddings miss valuable textual information, leading to suboptimal performance.
- Text-based (language model) approaches often fail to preserve logical structure.

This work introduces **OnT**, an ontology embedding method that tunes Pretrained Language Models (PLMs) using geometric modeling in hyperbolic space. It:

- Incorporates textual labels
- Simultaneously preserves class hierarchies and logical relationships of Description Logic EL
- Outperforms baselines (including state-of-the-art) in prediction & inference of axioms
- Shows robust transfer learning and application in discovering new axioms for SNOMED CT

Code: https://github.com/HuiYang1997/OnT

1. Introduction

- **OWL ontologies** (Web Ontology Language) use Description Logic (DL) axioms for explicit, formal, and shared domain knowledge.
- Examples: Gene Ontology (GO) in bioinformatics; SNOMED CT in healthcare.
- **Ontology embeddings:** Encode entities (concepts, roles, instances) as numerical vectors, preserving their structure and semantics for downstream tasks (prediction, inference, retrieval).

Two Main Paradigms

1. Geometric Model-Based Methods

- Encode ontology entities as geometric objects (instances = points, concepts = areas).
- **Examples:**
  - ELEM (Concepts as balls)
  - BoxEL, Box2EL, TransBox (Concepts as boxes)
- Translate DL operators into geometric operations:
  - Subsumption ⟶ area inclusion
  - Conjunction ⟶ intersection
- **Limitations:** Ignore textual/entity labels; can't embed unseen entities; limited in dynamic/transfer settings.

2. Language Model-Based Methods

- Encode textual content; transform axioms/graph structure to sentences for PLM tuning.

- **Examples:** OPA2Vec, OWL2Vec*
- Use word embeddings (Word2Vec) or Transformer-based PLMs.
- Capture text and formal semantics; higher similarity for related entities.
- **Limitations:** Ignore logical relationships; most approaches only use non-contextual embeddings, few use BERT-style PLMs, and have limited support for complex logical structures.
- **HiT**: Combines PLM and geometric modeling, but only for taxonomies (concept hierarchies), not complex logics.

Motivation & Solution

- **OnT:** Integrates strengths of PLMs for textual embedding with geometric modeling (hyperbolic space) for logical structure embedding. Embeds hierarchies & logical relationships (EL), improves axiom inference & supports new entities.

2. Related Work

Geometric Model-Based Methods

- Geometric objects for concepts/instances & specific geometric relations for roles (transition functions).
- Boxes, balls, cones, fuzzy sets for concepts.
- **Box-based methods:** Preferred since intersection of boxes yields another box (closure property); naturally handle conjunctions.
- Primarily focus on EL ontologies; some (catE, FALCON) support ALC but not widely used.

Language Model-Based Methods

- Word2Vec-based (OPA2Vec, OWL2Vec*) fine-tune embeddings for ontology entities, then use these for prediction (binary classifier).
- Recent work explores PLMs (Transformers); most approaches fine-tune models for specific tasks, not for general-purpose embeddings.
- Can't capture logical structures like transitivity within vector space.
- **HiT**: Hierarchy-encoder using PLMs and hyperbolic embedding for taxonomies; misses role embeddings and complex concept formation logic.

Exclusion

- Knowledge Graph completion methods (KG-BERT, KEPLER) excluded due to fundamental difference (relational triples vs. conceptual DL).

3. Preliminary

3.1 Ontology

- OWL ontologies represent concepts (unary predicates) and roles (binary predicates).
- Focus on **EL ontologies** (balance expressiveness & efficiency).
- Use sets: Concepts (NC), Roles (NR), Individuals (NI).
- EL-concepts recursively:
  - $\top \mid \bot \mid A \mid C \sqcap D \mid \exists r.C \mid \{a\}$
- **Axioms:**
  - TBox: $C \sqsubseteq D$
  - ABox: A(a), r(a, b)

Example 1

Atomic concepts: Teacher, Student, Class; roles: teach, hasClass, study; individuals: Dr.Smith, Emma

- TBox: Person$\sqcap\exists$teach.Class $\sqsubseteq$ Teacher
- Person$\sqcap\exists$study.Class $\sqsubseteq$ Student
- ABox: Teacher(Dr.Smith), hasClass(Emma, Math101)

Normalization

- Focus on TBox; ABox can be converted to TBox by treating instances as classes.
- **Normal Forms (NF1-NF4):**
  - $A \sqsubseteq B$
  - $A1 \sqcap A2 \sqsubseteq B$
  - $A \sqsubseteq \exists r.B$
  - $\exists r.B \sqsubseteq A$

- All EL-ontologies can be normalized (introduce new atomic concepts).

Example 2 (Normalization)

- Original: Person⊓∃teach.Class ⊑ Teacher
- Normalized:
  - Person⊓N1 ⊑ Teacher
  - N1 ⊑ ∃teach.Class
  - ∃teach.Class ⊑ N1 (N1 interprets as “Something that teaches some Class.”)

Inference

Interpretation I = (ΔI, ·I):

- Maps concepts/roles/individuals to sets and pairs over ΔI
- C ⊓ D ⟶ CI ∩ DI
- ∃r.C ⟶ all a in ΔI with a related to some b in CI by r
- I satisfies TBox axiom X ⊑ Y if XI ⊆ YI
- Entailment: O |= α if α is satisfied in all models of O

3.2 Hyperbolic Space

- Manifold M (locally Euclidean Rd) with Riemannian metric dM(x, y): Distance function
- Hyperbolic space Hn: Negative curvature −κ
- **Poincaré ball model:** Defines ball Bn with radius 1/√κ
- Distance:

  $$ d_κ(x, y) = \frac{1}{\sqrt{κ}} \arcosh\left(1 + \frac{2κ|x - y|^2}{(1 - κ|x|^2)(1 - κ|y|^2)}\right) $$

- Scaling:

  $$ k \odot x = \tanh(k \cdot \tanh^{-1}(|x|)) \cdot \frac{x}{|x|} $$

4. Methodology

OnT main steps:

1. **Concept Embedding:**
   - Embed atomic/complex concepts as points in hyperbolic space using PLMs and verbalization.
2. **Role Embedding:**
   - Roles as rotations and scaling in hyperbolic space. Captures logical patterns of existential qualifications ∃r.
3. **Training with Hierarchy:**
   - Use Poincaré ball model; axioms as hierarchical partial order.
   - Specialized loss functions for existential and conjunction.

4.1 Verbalisation-based Concept Embedding

- Each atomic concept A and role r has associated name/definition (V(A), V(r)).
- Systematic rule to verbalize complex concepts C:
  - V(C⊓D) = “V(C) and V(D)”
  - V(∃r.C) = “something that V(r) some V(C)”
  - Example: V(Person⊓Student) = "person and student"
- Embedding done by LM (BERT, mean pooling), then re-trained in hyperbolic space, result denoted as x_C

4.2 Logic-aware Role Embedding

- Role does not get direct embedding in verbalization; hard to preserve deductive patterns (e.g. monotonicity of ∃r).
- Solution: **Role embedding as function fr over hyperbolic space**
  - For ∃r.D, represent as fr(x_D) and encourage equivalence to x_∃r.D via extra loss term
  - fr defined as rotation/scaling:

```
fr(v) = k_r ⊙ (R(θ_r) · v)
```

where k_r (scaling), θ_r (rotation angles), and v ∈ H^{2m}.

- Rotation matrix R(θ_r) is block-diagonal with 2D rotation matrices.

4.3 Training

Hierarchy Loss

- Subsumption C ⊑ D encoded as x_C < x_D (partial order) in Poincaré model
- **Loss function:**
  - Contrastive loss: Brings related concept embeddings closer, separates negatives

    ```
    L_{contrast}(x_C < x_D) = max(0, d_κ(x_C, x_D) − d_κ(x_C, x_{Dneg}) + α)
    ```

  - Centripetal loss: Pulls parents (D) closer to origin than children (C)

    ```
    L_{centri}(x_C < x_D) = max(0, ‖x_D‖_κ − ‖x_C‖_κ + β)
    ```

  - Overall:

    ```
    L_{<}(x_C < x_D) = L_{contrast} + L_{centri}
    ```

- **Role Embedding Loss:** Align x_∃r.D and fr(x_D) using hierarchy loss (not direct Euclidean or hyperbolic distance)
  - Interpret equivalence by partial-orders both directions

    ```
    L_r(∃r.D) = 1/2[ L_{<}(x_∃r.D < fr(x_D)) + L_{<}(fr(x_D) < x_∃r.D) ]
    ```

- **Conjunction Loss:** For conjunction C⊓D, universally valid axiom
  - Use hierarchy loss to enforce that C⊓D ⊑ C and C⊓D ⊑ D

    ```
    L_⊓(C⊓D) = 1/2[ L_{<}(x_{C⊓D} < x_C) + L_{<}(x_{C⊓D} < x_D) ]
    ```

Training & Evaluation

- Total loss = sum of losses above (for all C⊑D, ∃r.D, C⊓D in ontology).
- Axiom evaluation score (confidence):

  ```
  s(C⊑D) = −[ d_κ(x_C, x_D) + λ(‖x_D‖_κ − ‖x_C‖_κ) ]
  ```

  λ optimized on validation.
- **Proposition 1:** For role function fr, with rotation/scaling, scores and distances can be made identical under certain settings (i.e., kr = 1).

5. Evaluation

5.1 Experiment Setting

- Main tasks: **Axiom prediction** and **Inference** (missing axioms, logical derivation)
- **Datasets:** GALEN, GO, Anatomy (Uberon). Used normalized EL version.
- Prediction task: 80/10/10 random split on axioms.
- Inference task: train on whole ontology; test on all inferred axioms (NF1 type); 1000 random validation subsumptions.

Dataset Statistics (Table: Train/Val/Test axiom counts)

Dataset	NF1	NF2	NF3	NF4	Total	Inferred NF1
GALEN	25610/3200/3203	11679/1459/1462	25299/3161/3165	6287/785/788	68875/8605/8618	335002
GO	116751/14593/14596	24097/3011/3014	238899/29861/29865	81948/10243/10245	461695/57708/57720	1184380
Anatomy	41764/5220/5222	12336/1542/1543	39766/4970/4972	7586/947/951	101452/12679/12688	225330

Baselines Compared

- **Geometric:** Box2EL, BoxEL, TransBox, ELBE, ELEM
- **Language Model-Based:** HiT, OPA2Vec, OWL2Vec*
- **OnT(w/o r):** OnT variant without role embeddings.
- Other task-specific methods (BERTSub, etc.) excluded.

Metrics

- Hits@k (H@k), Mean Reciprocal Rank (MRR), Mean Rank (MR)

Protocol

- Used all-MiniLM-L12-v2 (33.4M) as LM for OnT/HiT.
- Trained 1 epoch, 1 negative sample per axiom.
- Embeddings: average pooling over final LM layer.
- Θ(r), kr (role params): extra linear transformation on r embedding.
- Margins/learning rates fixed; λ chosen by validation.
- OPA2Vec/OWL2Vec*: use fine-tuned word embeddings + RF classifier (or Logistic Regression for GO).
- Other models reimplemented and retrained for consistency.

5.2 Prediction Task Results

- OnT consistently beats baselines in all datasets.
- Geometric methods get close H@k but much worse MR and MRR (poor average cases, many extreme/worst cases).
- OnT 7x better MR on GO.
- OPA2Vec/OWL2Vec* perform poorly with random forest classifier; HiT has better results due to hyperbolic structure.
- OnT's logical constraints + role embeddings improve both H@k and MR metrics.

5.3 Inference Task Results

- OnT leads all geometric methods, esp. on GO (3x better H@10/H@100, 5x better MR).
- HiT also strong, especially outside GO (lacks non-taxonomic info).
- Role embeddings and logical losses critical for best results.

5.4 Other Results

Ablation Study (Language Model Choice)

- All-MiniLM-L6-v2, all-MPNet-base-v2 tested; larger model had best average (lower MR), but not always best in every metric.

Transfer Learning

- Train/test on different datasets for axiom prediction; OnT and HiT both transfer well, OnT better on MR and H@k.

Case Study (Real Ontology Construction)

- SNOMED CT examples:
  - Finds a missing subsumption: “Stomach structure ⊑ Digestive organ structure”
  - Finds an erroneous superclass: corrects “Bone structure of upper limb” to better parent “Bone structure of extremity”

6. Conclusion & Future Work

- OnT combines geometric and text-based models for ontology embeddings; sets state-of-the-art on real ontologies for both prediction and inference, also enables transfer learning.
- Future: unify additional hierarchy embedding techniques, extend to richer ontologies (ALC with negation, role inclusion logic), analyze impact of verbalization quality and more datasets.
- All code/data available at https://github.com/HuiYang1997/OnT

References

(see full paper for all entries)

UmlsBERT: Clinical Domain Knowledge Augmentation of Contextual Embeddings

Authors

George Michalopoulos, Yuanxin Wang, Hussam Kaka, Helen Chen, Alex Wong
University of Waterloo
gmichalo, yuanxin.wang, hussam.kaka, helen.chen, alexander.wong @uwaterloo.ca

---

Abstract

Contextual word embedding models (BioBERT, Bio ClinicalBERT) achieve state-of-the-art in biomedical NLP by focusing pre-training on domain corpora.
But they lack expert domain knowledge integration.

Key contributions of UmlsBERT:

- Introduces domain knowledge during pre-training using UMLS.
- Uses two strategies:
    1. Connect words with the same underlying UMLS concept.
    2. Use UMLS semantic groups to build clinical input embeddings.
- Outperforms existing domain-specific models on named-entity recognition (NER) and clinical NLP inference tasks.

---

1. Introduction

- Healthcare data volume is enormous; advanced NLP is needed.
- NLP models, such as ELMo and BERT, show state-of-the-art results in general and biomedical domains after domain-specific pre-training.
- **Limitation:** Current biomedical transformer models exclude expert domain knowledge.
- **Unified Medical Language System (UMLS):**
    - Compendium of biomedical vocabularies; contains synonyms and hierarchies.
    - Connects terms by Concept Unique Identifiers (CUI).
        - E.g., "lungs" and "pulmonary" share CUI C0024109.
    - Groups concepts into semantic types (e.g., ANATOMY, DISORDER).
- **This work:**
    - Presents a new architecture (UmlsBERT) that integrates UMLS-based knowledge into the BERT pre-training phase for semantically enriched, clinically aware contextual representations.
    - Proposes a new multi-label loss for Masked LM, using CUI connections.
    - Adds semantic group embeddings.
    - Shows improved results over BioBERT, Bio ClinicalBERT, and domain BERT models in clinical NER and NLP tasks.

---

2. Related Work

2.1 Contextual Word Embeddings

- **Traditional embeddings**: word2vec, FastText → fixed vector per word, regardless of context.
- **ELMo (Peters et al.)**: Introduces context-sensitive embeddings using bidirectional language models.
- **BERT (Devlin et al.)**: Uses bidirectional transformer architecture. Pre-training on large corpora, then fine-tuned for tasks.

Knowledge-augmented contextualization:

- **Sense-BERT:** Predicts supersenses (semantic class) using WordNet.

- **GlossBERT:** Improves word sense disambiguation with context-gloss pairs.
- **LiBERT:** Uses synonym and hypernymy/ hyponymy pairs in additional pre-training tasks.

2.2 Contextual Clinical Embeddings

- Focus: performance in biomedical domain.
- **BioBERT:** Pre-trained BERT on biomedical corpora (PubMed, PMC).
  - Improves over general BERTs in biomedical NLP tasks.
- **Clinical adaptions:** Training or further pre-training on clinical notes improves outcomes (i2b2, MIMIC-III, etc.).
- **Bio ClinicalBERT:** Further pre-trained on clinical text (MIMIC-III).
  - Clinical context improves entity recognition and related tasks.

---

3. Data

- **Pre-training:** Used MIMIC-III database (Beth Israel Deaconess; 40k+ patients, anonymized intensive care records).
  - Focused on "NOTEEVENTS" — 2M+ rows of notes and diagnostic reports.
- **Evaluation:** Used MedNLI NLI task and four i2b2 NER tasks (2006 de-ID, 2010 concept extraction, 2012 and 2014 entity extraction/de-ID).
- **Datasets Table:**

Dataset	Train	Dev	Test	Classes
MedNLI	11232	1395	1422	3
i2b2 2006	44392	5547	18095	17
i2b2 2010	14504	1809	27624	7
i2b2 2012	6624	820	5664	13
i2b2 2014	45232	5648	32586	43

  - Used splits from Alsentzer et al. (2019).

---

4. Methods

4.1 BERT Model Recap

- **BERT:** Multi-layer bidirectional transformers. Captures meaning by context.
- **Pre-training tasks:**
  - **Masked Language Modeling (MLM):** Predict random masked tokens.
  - **Next Sentence Prediction:** (omitted in UmlsBERT)
- **Input embedding equation:**

```
u_j_input = p_j + SEG_seg_jid + E_wj
```

- Where:
  - $$p_j$$ = position embedding
  - $$SEG_{segjid}$$ = segment embedding (all ones for single sentence setups)
  - $$E_{wj}$$ = token embedding
- **Output:** Score vector for masked token.
- **Cross-entropy loss:**

```
loss = -log(exp(y_w_w) / sum(exp(y_w_w')))
```

4.2 Enhancing Contextual Embeddings with Clinical Knowledge

Semantic Group Embeddings

- Introduces new embedding matrix: $$SG \in \mathbb{R}^{d \times D_s}$$, $$D_s = 6$$ (unique UMLS semantic groups).
- For each clinical term, extract semantic group and embed with vector.

- **Input embedding update:**

```
u_j_input = u_j_input + SG_s(w)
```

- Non-UMLS terms use zero vector.
- **Goal:** Make embeddings of same-group words more similar.
  - Helps rare-word representation.

Updating MLM Loss with CUI Connections

- **Loss function adaptation:**
  - From single-label (one-hot) to multi-label (binary) vector indicating all words with the same CUI.
  - **Binary cross-entropy:**

```
loss = sum_over_D [ h_wi * log(y_wi) + (1-h_wi) * log(1-y_wi) ]
```

- **Result:**
  - Model learns underlying clinical relations.
- **Example:**
  - For masked “lungs”, original BERT predicts only "lungs"; UmlsBERT predicts all: "lungs", "lung", "pulmonary" (CUI C0024109).

4.3 UmlsBERT Training

- **Steps:**
  1. Initialize with Bio ClinicalBERT weights.
  2. Further pre-train on MIMIC-III with the updated masked LM.
  3. For downstream tasks, add linear layer, fine-tune on task-specific data (with word/CLS token embeddings).
  4. Used WordPiece tokenization.
- **Hyperparameters:**
  - 150,000 steps
  - batch size 32
  - max seq. length 128
  - learning rate $$5 \times 10^{-5}$$
  - All other settings default.
- **Hardware:** 2x nVidia V100 16GB GPUs, 224GB RAM, Ubuntu 18.04.3.
- **Pre-training time:** 5 days
- **Notes:**
  - UMLS chosen for:
    1. Broad coverage (MeSH, ICD-10).
    2. Could be generalized outside medical domain (e.g., with WordNet).

---

5. Results

5.1 Downstream Clinical NLP Tasks

- Compared BERT, BioBERT, Bio ClinicalBERT, UmlsBERT on MedNLI and i2b2.
- **Implementation:** Huggingface transformers (PyTorch 0.4.1).
- **TPUs:** Tesla P100 16.3GB, 32GB RAM, Ubuntu 18.04.3.
- **Hyperparameter tuning:**

- Batch size: 32–64.
- Learning rate: $$2 \times 10^{-5}$$, $$3 \times 10^{-5}$$, $$5 \times 10^{-5}$$.
- NER tasks: 20 epochs.
- MedNLI: 3–4 epochs.

Model	MedNLI Test Acc.	i2b2 2006 F1	i2b2 2010 F1	i2b2 2012 F1	i2b2 2014 F1	Avg Run time (sec)
BERTbase	77.9 ± 0.6	93.5 ± 1.4	85.2 ± 0.2	76.5 ± 0.2	95.2 ± 0.1	varies
BioBERT	82.2 ± 0.5	93.3 ± 1.3	87.3 ± 0.1	77.8 ± 0.2	94.6 ± 0.2	varies
Bio Cin. BERT	81.2 ± 0.8	93.1 ± 1.3	87.7 ± 0.2	78.9 ± 0.1	94.3 ± 0.2	varies
UmlsBERT	82.2 ± 0.1	93.4 ± 1.2	88.3 ± 0.2	79.3 ± 0.1	94.7 ± 0.1	varies

- **Random seeds:** 6809, 36275, 5317, 82958, 25368
- **Findings:**
  - UmlsBERT achieves best F1 in i2b2 2010/2012, best accuracy in MedNLI.
  - BERTbase best in i2b2 2006/2014 (due to different PHI masking procedures).
  - In all tasks, UmlsBERT outperforms other biomedical BERTs.

5.2 Qualitative Embedding Comparisons

- Nearest-neighbor evaluation for words in three categories:
  - **ANATOMY, DISORDER, GENERIC**
- Example Table:

Term	BERTbase	BioBERT	Bio Clin. BERT	UmlsBERT
heart	cardiac, lung	liver, lung	liver, lung	heartbeat, liver
kidney	mass, bleeding	liver, lung	Ren, masses	liver, Ren
mass	massive, sweating	weight, strokes	weight, bloody	lump, masses
bleeding	university, conflict	schooling, battle	bloody, university	hem, university
school	college, battle	college, battle	college, warfare	university, wartime
war	college, battle	college, warfare	college, warfare	wartime, hem

- **Findings:** Only UmlsBERT connects clinical synonyms via CUI (e.g., "kidney" ⟷ "Ren"). Generic term associations remain intact.

5.3 Semantic Group Embedding Visualization

- **t-SNE plot:**
  - UmlsBERT input embeddings cluster by semantic group.
  - Bio ClinicalBERT does not cluster by group.
  - Shows explicit semantic organization from UMLS augmentation.

---

6. Conclusion and Future Work

- **UmlsBERT:** Integrates domain clinical knowledge into contextual embedding pre-training.
- Demonstrates:
  - Learns groupings among clinically similar words.
  - Improves quality and performance over prior domain BERTs.
  - More meaningful input embeddings due to semantic group information.
- **Future:**
  - Test with more complex downstream architectures.
  - Use broader UMLS semantic group coverage.
  - Apply to larger models (BERT-large).

References

Key works cited include:

- Alsentzer et al. (Bio ClinicalBERT, 2019)
- Bodenreider (UMLS metathesaurus, 2004)
- Bojanowski et al. (FastText, 2016)
- Devlin et al. (BERT paper, 2019)
- Johnson et al. (MIMIC-III, 2016)
- Lee et al. (BioBERT, 2019)
- Miller (WordNet, 1995)
- Peters et al. (ELMo, 2018)
- Vaswani et al. (Transformers, 2017)
- Wolf et al. (Huggingface Transformers, 2019)
- Wu et al. (WordPiece, 2016)
- van der Maaten & Hinton (t-SNE, 2008)

BioBERT: Pre-trained Biomedical Language Model for Biomedical Text Mining

Authors: Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang
Institutions: Korea University, Naver Corp

Abstract

Motivation

- Biomedical text mining is important due to the fast growth of biomedical documents.
- Natural language processing (NLP) advancements allow extraction of valuable information from biomedical texts.
- Deep learning models for NLP have improved biomedical text mining, but applying models trained on general texts to biomedical texts is suboptimal due to vocabulary and distribution differences.
- The study investigates adapting the BERT model to biomedical texts.

Results

- **BioBERT** is a pre-trained language model for the biomedical domain, trained on large biomedical corpora.
- Architecture is nearly the same as BERT across tasks.
- BioBERT outperforms BERT and previous state-of-the-art in biomedical named entity recognition (NER), relation extraction (RE), and question answering (QA):
  - NER: +0.62 F1
  - RE: +2.80 F1
  - QA: +12.24 MRR
- Pre-training on biomedical corpora enables better understanding of complex texts.

Availability

- Pre-trained weights: [github.com/naver/biobert-pretrained]
- Fine-tuning code: [github.com/dmis-lab/biobert]

Introduction

- Biomedical literature volume increases rapidly (~3,000 articles per day in peer-reviewed journals).
- Tools are needed for accurate extraction of information from literature.
- Deep learning improved biomedical NER (LSTM, CRF), RE, and QA models.
- Applying general domain word models (Word2Vec, ELMo, BERT) is limited due to domain word distribution differences.
- Biomedical models must be trained on specific biomedical corpora.

Approach

- **BioBERT:** Pre-trained using BERT weights, then further pre-trained on:
  - PubMed abstracts
  - PMC full-text articles
- Fine-tuned for three major tasks: NER, RE, and QA.
- Various pre-training strategies assessed using different corpus combinations.

Contributions

- BioBERT: First domain-specific BERT-based model pre-trained on biomedical text.
- Pre-training BERT on biomedical corpora boosts its performance.
- Outperforms previous models on NER, RE, and QA with minimal architectural changes.
- Datasets, pre-trained weights, and code are public.

Materials and Methods

BERT: Bidirectional Encoder Representations from Transformers

- Previous models (Word2Vec, GloVe) generated context-independent word representations.
- Newer models (ELMo, CoVe, BERT) produce context-dependent representations.
- BERT uses a masked language model objective. It is trained bidirectionally with transformers.
- Bidirectional context is crucial for representing natural language, especially biomedical terms.

Pre-Training BioBERT

- BERT pre-trained on English Wikipedia and BooksCorpus (general domain).
- Biomedical texts need biomedical corpora due to specialized vocabulary.
- BioBERT is pre-trained on:
  - PubMed abstracts: 4.5B words
  - PMC full-text articles: 13.5B words

Table 1: Text Corpora Used

Corpus	Number of Words	Domain
English Wikipedia	2.5B	General
BooksCorpus	0.8B	General
PubMed Abstracts	4.5B	Biomedical
PMC Full-text articles	13.5B	Biomedical

Table 2: Pre-training Combinations

Model	Corpus Combination
BERT	Wiki + Books
BioBERT PubMed	Wiki + Books + PubMed
BioBERT PMC	Wiki + Books + PMC
BioBERT Both	Wiki + Books + PubMed + PMC

- Tokenization: WordPiece method handles out-of-vocabulary words by splitting into subwords.
- Cased vocabulary (keeping upper/lower case) performed better.
- Used original BERT vocabulary for compatibility.

Fine-tuning BioBERT

- Can be applied to downstream tasks with minimal modification.
- Fine-tuning performed for:
  - **NER:** Recognizes biomedical-specific entities. Evaluated using entity-level precision, recall, F1. Learns domain WordPiece embeddings directly.
  - **RE:** Classifies relations between named entities. Uses [CLS] token for classification. Entities anonymized (e.g., GENE/DISEASE).
  - **QA:** Answers questions given related texts. Uses the same BERT QA architecture as SQuAD.

Results

Datasets

NER Datasets (Table 3):

Dataset	Entity Type	Annotations
NCBI Disease	Disease	6,881
2010 i2b2VA	Disease	19,665
BC5CDR	Disease	12,694
BC5CDR	Drug/Chemical	15,411
BC4CHEMD	Drug/Chemical	79,842
BC2GM	Gene/Protein	20,703
JNLPBA	Gene/Protein	35,460
LINNAEUS	Species	4,077
Species-800	Species	3,708

RE Datasets (Table 4):

Dataset	Entity Type	Number of Relations
GAD	Gene-disease	5,330
EU-ADR	Gene-disease	355
CHEMPROT	Prot.-chem.	10,031

QA Datasets (Table 5):

Dataset	Number Train	Number Test
BioASQ 4b	327	161
BioASQ 5b	486	150
BioASQ 6b	618	161

Experimental Setups

- **Pre-training:**
  - BERTBASE on Wikipedia + BooksCorpus: 1M steps
  - BioBERT v1.0 (PubMed + PMC): 470K steps (200K on PubMed, 270K on PMC optimal)
  - BioBERT v1.1 (PubMed): 1M steps
- **Hardware:**
  - Pre-training: 8×NVIDIA V100 32GB GPUs (10 to 23 days)
  - Fine-tuning: 1×NVIDIA Titan Xp 12GB GPU (efficient; less than 1 hour for RE and QA tasks)
  - Fine-tuning parameters: batch size 10/16/32/64; learning rates 5e-5, 3e-5, 1e-5

Experimental Results

NER Results (Table 6 Highlight)

Dataset	Metric	SOTA	BERT	BioBERT v1.0	BioBERT v1.1
NCBI Disease	F1	88.6	85.63	87.79	**89.71**
BC5CDR Disease	F1	86.23	82.41	85.27	**87.15**
BC2GM	F1	81.69	81.79	83.53	**84.72**
...	...	...	...	...	...

- BioBERT outperforms BERT and prior state-of-the-art in six of nine datasets.

RE Results (Table 7 Highlight)

Dataset	Metric	SOTA	BERT	BioBERT v1.0	BioBERT v1.1
GAD	F1	83.93	79.29	80.24	79.83
EU-ADR	F1	85.34	84.62	86.51	79.74
CHEMPROT	F1	64.10	73.74	75.13	**76.46**

- BioBERT has the highest F1 on 2 of 3 biomedical RE datasets.

QA Results (Table 8 Highlight)

Dataset	Metric	SOTA	BERT	BioBERT v1.1
BioASQ 4b	MRR	23.52	33.77	**34.72**
BioASQ 5b	MRR	47.24	44.27	**51.64**
BioASQ 6b	MRR	27.84	40.88	**48.43**

- BioBERT sets new state-of-the-art on all biomedical QA datasets for MRR.

Discussion

- Varying PubMed corpus size for pre-training showed most performance improvements up to 4.5B words.
- More pre-training steps generally improved NER results.
- BioBERT can identify named entities BERT cannot and provides more accurate QA answers.

Conclusion

- Pre-training language models on biomedical corpora is essential for biomedical text mining tasks.
- BioBERT provides large performance gains with almost no architectural changes.
- Resources and code are public for the community.
- New versions using only PubMed or custom biomedical vocabulary are planned.

Funding

- Supported by National Research Foundation of Korea (NRF-2017R1A2A1A17069645, NRF-2017M3C4A7065887, NRF-2014M3C9A3063541).

References

List includes works on BERT, Word2Vec, ELMo, SQuAD, and important biomedical datasets and text mining tasks. Not reproduced here due to length, but should be included in any full reproduction.

Note: Every numerical value, summary statement, definition, and example is retained as in the original source, using Markdown structure for clarity and usability. Tables are rendered as markdown as shown.[1]

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin
NIPS 2017 (arXiv:1706.03762)

Abstract

- Traditional sequence transduction models use recurrent or convolutional neural networks with attention for tasks like machine translation.

- The paper presents **Transformer**, a novel network architecture based only on attention, without any recurrence or convolution.
- On translation tasks (English-German and English-French, WMT 2014 benchmarks), the Transformer outperforms all previous models, including ensembles, and trains much faster.
- The model also generalizes well to other sequence tasks, such as English constituency parsing.

---

1. Introduction

- **Sequence modeling and transduction** tasks (like NMT) have long used RNNs, particularly LSTMs and GRUs.
- Recurrent models operate sequentially: output at time $$ t $$ depends on outputs at previous steps, limiting parallelization.
- New models (factorized RNNs, conditional computation) improve computation but retain sequential bottlenecks.
- **Attention mechanisms** allow direct modeling of dependencies between positions, improving context access, but were mainly used with RNNs.
- **Transformer** removes recurrence entirely, using only attention to relate input and output positions, enabling parallelization and faster training.

---

2. Background

- Prior methods to reduce sequential operations include ConvS2S and ByteNet, using convolutions for parallelism, but relating distant positions is expensive (linearly or logarithmically with distance).
- In Transformer, dependencies between any positions require only a constant number of operations.
- **Self-attention (intra-attention):** attention applied to positions within a single sequence, used for tasks like reading comprehension, summarization, and sentence representation.
- Transformer is the first model to use only self-attention for sequence transduction without RNNs or convolutions.

---

3. Model Architecture

Encoder-Decoder Structure

- The model uses the standard encoder-decoder structure.
  - **Encoder:** maps input symbols $$ x_1, ..., x_n $$ to continuous representations $$ z_1, ..., z_n $$.
  - **Decoder:** outputs a sequence $$ y_1, ..., y_m $$, one symbol at a time.

Encoder

- Stack of $$ N = 6 $$ identical layers.
- Each layer:
  1. Multi-head self-attention layer.
  2. Position-wise fully connected feed-forward layer.
- **Residual connection** and **layer normalization** after each sub-layer:

```
Output = LayerNorm(x + Sublayer(x))
```

- All outputs (sub-layers and embeddings) have dimension $$ d_{\text{model}} = 512 $$.

Decoder

- Also a stack of $$ N = 6 $$ identical layers.
- Each layer has three sub-layers:
  1. Masked multi-head self-attention (prevents attending to future positions).
  2. Multi-head attention over encoder output.
  3. Feed-forward network.
- Residual connection and layer normalization are applied.

3.2 Attention

Attention Function

- Maps a **query** and set of **key-value pairs** to an output.

- Output: weighted sum of values, weights from a compatibility function of query to each key.

3.2.1 Scaled Dot-Product Attention

- Given:
  - Queries $$ Q $$, keys $$ K $$, values $$ V $$, all as matrices.
  - Dimensions: Queries/Keys $$ d_k $$, Values $$ d_v $$.
- Calculation: $$ \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q K^T}{\sqrt{d_k}}\right) V $$
- Scaling (by $$ \sqrt{d_k} $$): prevents softmax from having small gradients for large $$ d_k $$.

3.2.2 Multi-Head Attention

- Instead of one attention operation, do $$ h $$ in parallel (with independent projections).
- Project $$ Q, K, V $$ $$ h $$ times to size $$ d_k, d_k, d_v $$ each, run attention, then concatenate and project:
  - For $$ \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) $$
  - Output: $$ \text{Concat}(\text{head}_1, ...,\text{head}_h)W^O $$
- Allows capturing information from different subspaces.
- In Transformer: typically $$ h = 8 $$, $$ d_k = d_v = d_{\text{model}} / h = 64 $$.

3.2.3 Use in Model

- Three uses in Transformer:
  - **Encoder-Decoder Attention**: decoder queries encoder outputs.
  - **Encoder Self-Attention**: attends over positions in input.
  - **Decoder Self-Attention**: attends over previous positions in output (using masking for causality).

3.3 Position-wise Feed-Forward Networks

- Each encoder/decoder layer has a feed-forward network applied identically at every position:

```
FFN(x) = max(0, xW1 + b1) W2 + b2
```

- Two linear transforms (can be seen as conv1d with kernel size 1); inner layer $$ d_{ff} = 2048 $$.

3.4 Embeddings and Softmax

- Input/output tokens embedded into vectors ($$ d_{\text{model}} $$).
- Output from decoder transformed and softmaxed to produce probabilities.
- Sharing weights: embedding and pre-softmax weight matrices are shared/scaled.

3.5 Positional Encoding

- No recurrence/convolution means position must be injected manually.
- **Sinusoidal positional encodings** added to input embeddings: $$ PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \ PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) $$
- Covers a range of positions and frequencies; enables relative/absolute positioning.

---

4. Why Self-Attention

- Compare self-attention, RNN, and CNN approaches for:
  - Complexity per layer
  - Parallelization (min sequential operations)
  - Path length between positions (for long-range dependencies)

Complexity, Sequential Operations, Max Path Length

Layer Type	Complexity	Sequential Ops	Max Path Length
Self-Attention	$$O(n^2 d)$$	$$O(1)$$	$$O(1)$$

Layer Type	Complexity	Sequential Ops	Max Path Length
Recurrent	$$O(n d^2)$$	$$O(n)$$	$$O(n)$$
Convolutional	$$O(k n d^2)$$	$$O(1)$$	$$O(\log_k n)$$

- Self-attention is more parallelizable; path length for dependencies is shortest.
- For large sequences, self-attention's cost can be limited by restricting attention to neighborhoods.

---

5. Training

5.1 Data and Batching

- **EN-DE:** WMT 2014 set, ~4.5M pairs, byte-pair encoding, 37k tokens.
- **EN-FR:** WMT 2014 set, 36M pairs, 32k word-piece vocab.

5.2 Hardware and Schedule

- Trained on 8x P100 GPUs.
- **Base model:** 0.4 sec/step, 100k steps (12 hours).
- **Big model:** 1.0 sec/step, 300k steps (3.5 days).

5.3 Optimizer

- **Adam** ($$\beta_1 = 0.9$$, $$\beta_2 = 0.98$$, $$\epsilon = 10^{-9}$$).
- Learning rate schedule: $$ \text{lrate} = d_{\text{model}}^{-0.5} \ \min\left(\text{step_num}^{-0.5}, \text{step_num} \cdot \text{warmup_steps}^{-1.5}\right) $$
- Warmup steps = 4000.

5.4 Regularization

- **Dropout:** 0.1 rate.
- **Label smoothing:** value 0.1.

---

6. Results

6.1 Machine Translation

Model	EN-DE BLEU	EN-FR BLEU	Training Cost (FLOPs)
ByteNet	23.75	-	1e20
GNMT RL	24.6	39.9	2.3e19/1.4e20
ConvS2S	25.2	40.5	9.6e18/1.5e20
Transformer (base)	27.3	38.1	3.3e18
Transformer (big)	28.4	41.8	2.3e19

- Transformer beats all previous results.
- **BLEU**: translation quality score.

6.2 Model Variations

- Single-head attention is worse than multi-head (best is h=8).
- Reducing key size ($$d_k$$) hurts; big models are better; dropout helps avoid overfitting.
- Sinusoidal vs learned positional encoding: similar results.

6.3 English Constituency Parsing

Model	WSJ Only F1	Semi-supervised F1
Transformer	91.3	92.7

- Transformer achieves high F1 (accuracy) on parsing tasks, competitive with or better than prior state-of-the-art.

---

7. Conclusion

- Introduced Transformer: first sequence model relying solely on attention, dropping recurrence and convolution.
- Achieves state-of-the-art on machine translation and generalizes to other tasks, with much faster training.
- Future work: multi-modal input, local/restricted attention for large inputs, reducing sequentiality further.

Appendix: Attention Visualizations

- Attention heads in various layers capture:
  - Long-distance grammatical relationships.
  - Anaphora/coreference resolution.
  - Structural dependencies in language.

Examples were visualized, showing:

- Attention on verbs linked to distant objects (e.g., "making ... more difficult").
- Sharp attention for possessive pronouns ("its").
- Different heads discovering different syntactic/semantic roles.

References

(The original paper includes ~40 technical references; see original for detail.)

Key Equations/Blocks

Scaled Dot-Product Attention

```
\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q K^T}{\sqrt{d_k}}\right) V
```

Position-wise Feed Forward

```
\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2
```

Positional Encoding

```
PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)
PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)
```

Learning Rate

```
\text{lrate} = d_{\text{model}}^{-0.5} \ \min(\text{step\_num}^{-0.5}, \text{step\_num} \cdot
\text{warmup\_steps}^{-1.5})
```

(Tables and figures referenced in the full paper were summarized in text or Markdown tables. For strict study use, refer to the original PDF for exact numerical tables and figures.)[1]

LLMonFHIR: A Physician-Validated, LLM-Based Mobile Application for Querying Patient Electronic Health Data

---

Authors and Affiliations

- Paul Schmiedmayer, PhD, Adrit Rao, Philipp Zagar, Lauren Aalami, Vishnu Ravi, MD, Aydin Zahedivash, MD, MBA, Dong-han Yao, MD, Arash Fereydooni, MD, Oliver Aalami, MD
- Stanford Mussallem Center for Biodesign, Stanford University, Stanford, California, USA
- Departments of Pediatrics, Emergency Medicine, Surgery, Stanford University

---

Abstract

Background

- Federal laws require interoperability of Electronic Health Records (EHRs) through Fast Healthcare Interoperability Resources (FHIR) APIs.
- Barriers exist for patient EHR access: limited functionality, English literacy, and health literacy.

Objectives

- Develop and evaluate a digital health solution to improve patient engagement with personal health data, focusing on chronic cardiovascular patients.

Methods

- Developed LLMonFHIR, an open-source mobile app using large language models (LLMs) enabling natural language interaction with health records across complexity levels, languages, and with bidirectional text-to-speech.
- Physicians evaluated responses to queries on 6 SyntheticMass FHIR patient datasets, scoring accuracy, understandability, and relevance on a 5-point Likert scale.

Results

- 210 LLMonFHIR responses evaluated.
- High median scores: accuracy 5/5, understandability 5/5, relevance 5/5.
- Challenges: summarizing health conditions, retrieving lab results; response variability and occasional information omission noted, highlighting the need for data preprocessing.

Conclusions

- LLMonFHIR shows strong potential to empower individuals with limited functional, English, and health literacy to access and benefit from patient-accessible EHRs.

Keywords

- Artificial intelligence, digital health, large language model, literacy, mobile application

---

Introduction

Importance of EHR Access

- Patient access to EHRs supports autonomy and improves healthcare quality by enhancing communication, efficiency, adherence, safety, education, satisfaction, activation, and self-efficacy.
- The 21st Century Cures Act (2016) promotes standardized APIs for interoperability, specifically Fast Healthcare Interoperability Resources (FHIR).

Existing Barriers

- Despite legislation, barriers persist:
  - Low functional literacy (20% U.S. adults lack required literacy for basic tasks).
  - Over half of Americans aged 16-74 read below sixth-grade level.
  - Over one-third (77 million) have poor health literacy.
  - 8% speak English less than "very well."

    - 20% speak languages other than English at home.
- These factors limit effective patient engagement with personal health data.

Need for Innovative Solutions

- Current investments in EHR interoperability unequally benefit patients.
- LLMonFHIR aims to overcome these barriers by using LLMs to enable conversational health record queries.

LLMonFHIR Overview

Concept and Technology

- LLMonFHIR is an open-source iOS app using LLMs (OpenAI's GPT-4) to interactively access FHIR-standardized EHR data.
- Employs Retrieval Augmented Generation (RAG) with function calling to dynamically retrieve only relevant FHIR resources based on user queries, optimizing context window limits and response relevance.
- Supports multiple languages and bidirectional text-to-speech.

Technical Implementation

- Built with Swift using Stanford's open-source Spezi framework.
- Integrates with Apple HealthKit to access records from multiple certified EHR systems.
- LLM prompts and system configurations are open-source and publicly available.

Handling Large Data and Context Challenges

- LLM context windows limited, can only handle part of large datasets (e.g., 10,000+ resources).
- Without filtering, risk "lost in the middle" problem where LLM loses track of middle content.
- Uses filtering to include only active medications and most recent lab/observation data.
- Patient's overall record context injected to maintain comprehension.

Application User Interface

- **Overview Screen:** Lists all available FHIR resources for user browsing.
- **Translation and Localization:** Can translate and summarize records in patient-friendly language (e.g., German).
- **Interactive Chat:** Users ask questions about their health data and receive natural language responses generated by the LLM.

Physician Evaluation Study

Study Design

- Evaluated ability to retrieve accurate, understandable, relevant answers using synthetic patient data (Synthea SyntheticMass dataset).
- Selected 6 patient data sets representing different cardiovascular conditions.
- 5 physicians (specialties: vascular surgery, pediatrics, internal medicine, critical care) evaluated responses.
- 7 standard questions based on common clinical queries; repeated 5 times per patient for a total of 210 responses.

Standard Questions Evaluated

ID	Question
Q1	What are my current medications and how should I be taking them?
Q2	What are the most common side effects of my medications?
Q3	Am I allergic to any of my medications?
Q4	Can you summarize my current medical conditions?
Q5	What health behaviors should I incorporate daily to help with my conditions?
Q6	Can you summarize my current medical conditions in German?

ID	Question
Q7	What are my recent laboratory values, their meanings, and how can I improve them?

Scoring Method

- Each response rated on a 5-point Likert scale for accuracy, understandability, and relevance.
- Accuracy: correctness of information.
- Understandability: clarity and patient-friendliness of language.
- Relevance: inclusion of all required information without vague or tangential material.

Results

Scores Summary

- Medians for medication-related questions (Q1, Q2, Q3): accuracy 5, understandability 5, relevance 5.
- Medians for condition-related questions (Q4, Q5, Q6): generally high but some deductions due to inclusion of out-of-context information or overly generic advice.
- Lab values question (Q7) showed most variability with medians ~4-5, but scores ranged widely (0-5), indicating challenges in data retrieval and interpretation.

Strengths Noted

- Clear, concise medication information including dosage and route explanations.
- Appropriate caution in some medication contexts (e.g., insulin recommendations).
- Effective identification of allergies or absence thereof.
- Accurate translations with generally good preservation of meaning in German.

Limitations Noted

- Occasional missing dosage or administration details.
- Mis-association of medication use and conditions (e.g., insulin linked incorrectly to prediabetes).
- Inclusion of non-medical social history data in medical condition summaries.
- Variability in repeated responses with about 20% omissions of conditions.
- Some German translations lacked nuance or left out descriptive text.
- Inconsistent or missing lab results retrieval occasionally impacted response accuracy and relevance.

Discussion

Interpretation

- LLMonFHIR successfully translates complex health data into patient-friendly language.
- Supports multilingual interaction and bidirectional speech capabilities.
- Response variability indicates LLMs can deliver nuanced, context-tailored answers but also necessitates oversight.

Challenges

- Filtering and preprocessing of patient data critical to optimize context for LLM and reduce irrelevant or outdated content.
- Difficulty in temporal understanding: older conditions sometimes misidentified as recent.
- Variability in function calling leads to inconsistent data retrieval; planned improvements include date component inclusion to handle historical data.

Privacy and Deployment Considerations

- Moving LLM execution closer to user's device (fog/local computing) enhances privacy and trust vs. centralized cloud.
- Possibility of layered model use: small on-device summarizers combined with advanced cloud interpreters.

Impact on EHR Access

- Despite increased API usage in hospitals, EHR access inequities persist due to literacy and digital divides.
- LLMonFHIR addresses functional, English, and health literacy barriers but not digital literacy or device accessibility.

Study Limitations

- Small physician sample size may limit generalizability.
- Synthetic data may not capture full complexity of real-world EHRs.
- Response variability and occasional information omissions show preprocessing and function calling need refinement.
- LLMonFHIR iOS app can't overcome smartphone ownership and digital literacy barriers impacting some patient populations.

Conclusions

- LLMonFHIR is the first patient-facing application using LLMs to summarize, contextualize, explain, and translate health records via FHIR API.
- Combines RAG, function calling, and automated data filtering to optimize response speed, relevance, and cost efficiency.
- Uses Apple HealthKit aggregation for seamless multi-source data access on mobile devices.
- Though not solving all barriers, this physician-validated tool shows promise in empowering individuals with limited functional, English, and health literacy.
- Open-source code available for research and integration into digital health initiatives.

Data and Software Availability

- Software and prompts are open-source:
  - Stanford BDHG GitHub: https://github.com/StanfordBDHG
  - Stanford Spezi GitHub: https://github.com/StanfordSpezi

Acknowledgments

- Supported by the Stanford Byers Center for Biodesign and Mussallem Center for Biodesign.
- Authors declare no relevant conflicts of interest.

References

- Includes 29 key references covering patient access to records, LLMs in healthcare, FHIR standards, digital literacy statistics, and open-source software.

Appendices and Figures (Descriptions)

Central Illustration

- Depicts LLMonFHIR workflow: user queries processed by cloud-hosted LLM (GPT-4) using targeted function calls retrieving relevant FHIR data from EHR systems via Apple HealthKit, delivering AI-generated responses to mobile app interface.

Table 1: Synthetic FHIR Patient Datasets Summary

- Detailed patient profiles including sex, age, medical conditions, allergies, and medications illustrating diverse cardiovascular patient cases used in evaluation.

Table 2: Physician Evaluator Questions

- Lists the seven standardized questions posed to LLMonFHIR during evaluation.

Figure 1: User Interface Samples

- Overview of FHIR records, language localization, and interactive chat feature screenshots from LLMonFHIR app.

Figure 2: Physician Score Distribution

- Graph showing accuracy, understandability, and relevance Likert scores across 210 responses with median values at maximum.

Figure 3: Example LLMonFHIR Responses

- Illustrates examples showing strengths (accurate no allergies and medication explanations) and limitations (generic advice, incomplete lab results retrieval).

Competency Statements

Medical Knowledge

- LLMonFHIR improves patient comprehension of EHRs, especially for individuals with limited functional, health, and English literacy.

Patient Care

- Empowers patients managing chronic cardiovascular conditions by enabling multilingual and complexity-adjusted EHR interaction.

Translational Outlook

- Future work to validate in real-world patients, include temporal data processing improvements, extend to Android platforms, and advance privacy-preserving on-device LLM execution.

Summary

LLMonFHIR represents a pioneering effort in using large language models to break down barriers in patient access to electronic health information. It is a comprehensive mobile application connecting patients with their FHIR-standardized health records using natural language queries, multi-language support, and speech features. The physician-validated evaluation demonstrates promising accuracy and patient-friendliness with noted areas for improvement, particularly in data preprocessing and consistent response generation. As interoperability becomes pervasive, tools like LLMonFHIR are critical for equitable health literacy and patient empowerment.