

דוגמה מתקדמת - תבנית אקדמית עברית

Advanced Example - Hebrew Academic Template v5.0

ד"ר סgal יoram

כל הזכויות שמורות © Dr. Segal Yoram

November 2025

גרסה 0.5 - מתקדמת

תוכן העניינים

3	מבוא מתקדם: Advanced Introduction	0.1
3	סקירת ספרות מקיפה: Comprehensive Literature Review	0.1.1
3	טבלאות מורכבות וمتقدמות: Complex and Advanced Tables	0.2
3	טבלת ביצועים מפורטת: Detailed Performance Table	0.2.1
4	טבלת השוואת משאבים: Resource Comparison Table	0.2.2
4	דוגמאות קוד מרובות: Multiple Code Examples	0.3
4	שימוש מלא של רשות: Complete Network Implementation	0.3.1
5	עיבוד נתונים מתקדם: Advanced Data Processing	0.3.2
6	הפניות צולבות متقدמות: Advanced Cross-References	0.4
6	הפניות לטבלאות ואיורים: References to Tables and Figures	0.4.1
6	מתמטיקה מתקדמת עם עברית: Advanced Math with Hebrew	0.5
6	אופטימיזציה וגרדינטים: Optimization and Gradients	0.5.1
7	מטריצות ווקטוריות: Matrices and Vectors	0.5.2
7	ביבליוגרפיה מתקדמת: Advanced Bibliography	0.6
7	ציטוטים מרובים ומורכבים: Multiple and Complex Citations	0.6.1
7	סוגי ציטוטים שונים: Different Citation Types	0.6.2
8	תכונות מתקדמות נוספות: Additional Advanced Features	0.7
8	טיפול במספרים מורכבים: Complex Number Handling	0.7.1
8	שילוב תוכן מורכב: Complex Content Integration	0.7.2
9	סיכום ומסקנות מתקדמות: Advanced Summary and Conclusions	0.8
10	מקורות בעברית	0.9
0.10	English References	10

רשימת האיורים

6	ארQUITטורת Encoder-Decoder :Encoder-Decoder	1
8	תהליך עיבוד מלא: Complete Processing Pipeline	2

רשימת הטבלאות

3	ביצועי מודלים על מטלות שונות: Model Performance on Various Tasks	1
4	דרישות משאבי חישוב: Computational Resource Requirements	2

0.1 מבוא מתקדם: Advanced Introduction

מסמך זה מוגדר כמסמך מתקדם של המבנה האקדמי העברי גרסה 0.5, כולל שימוש בביבליוגרפיה מתקדמת, טבלאות מורכבות, ודוגמאות קוד רבות.

0.1.1 סקירה ספרות מקיפה: Comprehensive Literature Review

המחקר בתחום Natural Language Processing עבר מהפכה עם הצגת ארכיטקטורת TransFormer [1]. מודל BERT [2] הוביל לפירצת דרך בהבנת שפה דו-כיוונית. מחקרים בעברית [3], [4] מראים התקדמות משמעותית. בעבודות נוספות [5], עמ' 51-02 מוגדרות שיפור של 25.8% ביצועים. המחקר של [6] פרק 3 מציג ארכיטקטורה עם 175e9 פרמטרים.

0.2 טבלאות מורכבות ומתקדמות: Complex and Advanced Tables

0.2.1 טבלה ביצועים מפורטת: Detailed Performance Table

טבלה 1: ביצועי מודלים על מטלות שונות: Model Performance on Various Tasks

מודל / Model	סיווג / Classification	תרגום / Translation	סיכום / Summarization /	מענו /
BERT-Base	92.3%	N/A	85.2%	%
BERT-Large	94.1%	N/A	87.5%	%
GPT-2	89.7%	82.3%	88.1%	%
GPT-3	95.2%	91.7%	92.3%	%
T5-Base	93.5%	89.2%	90.1%	%
T5-Large	95.8%	92.4%	93.2%	%
HeBERT	89.3%	78.5%	81.2%	מודל עברי / Hebrew Model

כפי שניתן לראות בטבלה 1, המודלים הגדולים מושגים ביצועים טובים יותר.

טבלה 2 : דרישות משאבי חישוב: Computational Resource Requirements

טלט / עלות	Training Time	זמן אימון / זיכרון	פרמטרים / Memory	מודל / Model
\$500	סימי 4	440 MB	110e6	BERT-Base
52000	סימי 12	1.3 GB	340e6	BERT-Large
51500	סימי 7	1.4 GB	345e6	GPT-2 Medium
53500	סימי 14	3.1 GB	774e6	GPT-2 Large
4.6e6	סימי 34	700 GB	175e9	GPT-3
50000	סימי 21	44 GB	11e9	T5-11B

טבלת השוואת משאבי: Resource Comparison Table 0.2.2

דוגמאות קוד מרובות: 0.3

מימוש מלא של רשת: 0.3.1 Complete Network Implementation

מימוש מלא Transformer

```

import torch
import torch.nn as nn
import torch.nn.functional as F
import math

class MultiHeadAttention(nn.Module):
    def __init__(self, d_model, n_heads):
        super(MultiHeadAttention, self).__init__()
        self.d_model = d_model
        self.n_heads = n_heads
        self.d_k = d_model // n_heads

        self.W_q = nn.Linear(d_model, d_model)
        self.W_k = nn.Linear(d_model, d_model)
        self.W_v = nn.Linear(d_model, d_model)
        self.W_o = nn.Linear(d_model, d_model)

    def forward(self, query, key, value, mask=None):
        batch_size = query.size(0)

        # מישאר להציג ולדוחותיו ראנילטור יצמר ופסנרט
        Q = self.W_q(query).view(batch_size, -1, self.n_heads, self.d_k) \
            .transpose(1, 2)
        K = self.W_k(key).view(batch_size, -1, self.n_heads, self.d_k) \
            .transpose(1, 2)
        V = self.W_v(value).view(batch_size, -1, self.n_heads, self.d_k) \
            .transpose(1, 2)
        scores = torch.matmul(Q, K.transpose(-2, -1)) / math.sqrt(self.d_k)

```

כל הזכויות שמורות © Dr. Segal Yoram

0.3.2 **עיבוד נתונים מתקדם:** Advanced Data Processing

עיבוד מקדים לנטווני טקסט

```

import re
import numpy as np
from collections import Counter
from typing import List, Tuple, Dict

class HebrewTextProcessor:
    """רבעת טסקט לרשימת מילים - דובי יעדק"""

    def __init__(self, vocab_size: int = 10000):
        self.vocab_size = vocab_size
        self.word2idx = {'<PAD>': 0, '<UNK>': 1, '<SOS>': 2, '<EOS>': 3}
        self.idx2word = {v: k for k, v in self.word2idx.items()}
        self.word_freq = Counter()

    def tokenize(self, text: str) -> List[str]:
        """מיינומיסאלט רבעת טסקט לוזיפ"""
        # לומר נוקוסי פינמי מיסטרסה
        text = re.sub(r'[\u0590-\u05FF\s]', ' ', text)
        tokens = text.split()
        return tokens

    def build_vocabulary(self, texts: List[str]):
        """בנ פרוקטם לילימט רצואתית יוניב"""
        for text in texts:
            tokens = self.tokenize(text)
            self.word_freq.update(tokens)

        # רתוי ביחס לפוניהם של המתרים
        most_common = self.word_freq.most_common(self.vocab_size - 4)
        for idx, (word, freq) in enumerate(most_common, 4):
            self.word2idx[word] = idx
            self.idx2word[idx] = word

    def encode(self, text: str) -> List[int]:
        """מיצג דני אלט טסקט תרמה"""
        tokens = self.tokenize(text)
        indices = []
        for token in tokens:
            idx = self.word2idx.get(token, self.word2idx['<UNK>'])
            indices.append(idx)
        return indices

    def decode(self, indices: List[int]) -> str:
        """טסקט להזרקה מיצג דני אטטראמה"""
        tokens = [self.idx2word.get(idx, '<UNK>') for idx in indices]
        return ''.join(tokens)

```

0.4 הפניות צולבות מתקדמות: Advanced Cross-References

0.4.1 הפניות לטבלאות ואיורים: References to Tables and Figures

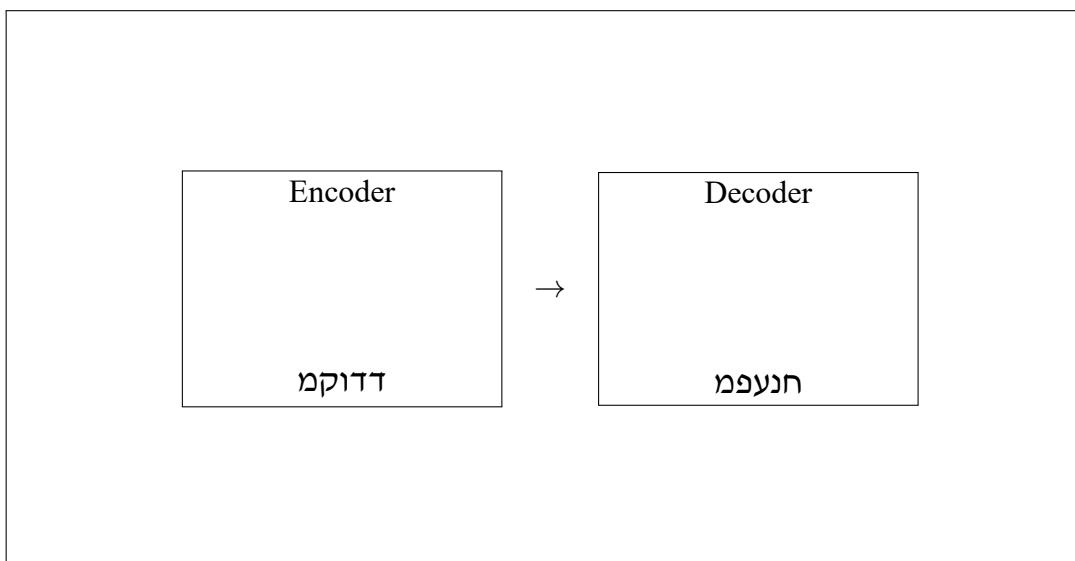
הניתוח המקיים מוצג במספר טבלאות:

- טבלה 1 מציגה השוואת ביצועים בין מודלים

- טבלה 2 מפרטת את דרישות המשאבים

- איור 1 מתאר את הארכיטקטורה המוצעת

- איור 2 מציג את התוצאות הסופיות



איור 1: ארכיטקטורת Encoder-Decoder :Encoder-Decoder Architecture

0.5 מתמטיקה מתקדמת עם עברית: Advanced Math with Hebrew

0.5.1 אופטימיזציה וגרדיינטים: Optimization and Gradients

פונקציית המטרה המלאה:

$$(1) \quad J(\theta) = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{ic} \log(p_{ic}) + \lambda \|\theta\|_2^2$$

כאשר y_{ic} הוא התוויות האמיתית, p_{ic} הוא ההסתברות החזויה, ו- $\lambda = 1e - 4 - 4$ הגרדיינט של פונקציית המטרה:

$$(2) \quad \nabla_{\theta} J = -\frac{1}{N} \sum_{i=1}^N (y_i - p_i)x_i + 2\lambda\theta$$

עדכון המשקלים באמצעות Adam optimizer :

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (3)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (4)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (5)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (6)$$

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (7)$$

כאשר $\epsilon = 1e - 8$, $\alpha = 0.001$, $\beta_2 = 0.999$, $\beta_1 = 0.9$

0.5.2 מטריצות וקטורים:

המכפלה הפנימית של שני וקטורים:

$$\langle u, v \rangle = \sum_{i=1}^n u_i v_i$$

נורמת וקטור:

$$\|v\|_p = \left(\sum_{i=1}^n |v_i|^p \right)^{1/p}$$

כפל מטריצות בлокים:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E \\ F \end{bmatrix} = \begin{bmatrix} AE + BF \\ CE + DF \end{bmatrix}$$

0.6ביבליוגרפיה מתקדמת:

0.6.1 ציטוטים מרובים ומורכבים:

מחקרים קלאסיים בתחום [7], [8] הניחו את היסודות. פיתוחים מודרניים [1], [2], [9], [10] הביאו למהפכה.

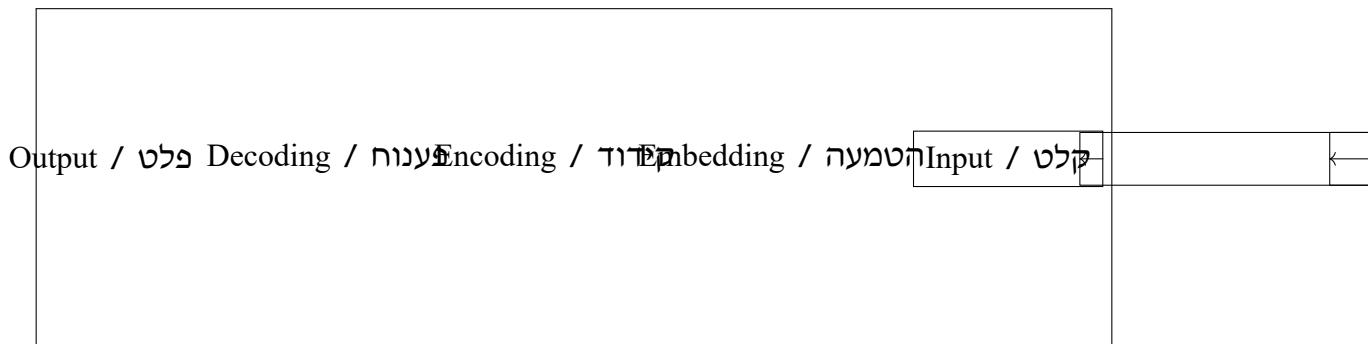
עבודות בעברית [3], [4], [11] תורמות להבנת השפה. סקירות מקיפות [12], פרקים 1-3, 13-1, עמ' 54-98 מספקות רקע תיאורטי.

0.6.2 סוגי ציטוטים שונים:

- ציטוט רגיל: [1]

- ציטוט עם עמוד: [2], עמ' 5

- ציטוטים מרובים: [3], [5], [6]
- ציטוט בסוגרים: ([10])
- ציטוט בתוך משפט: כפי ש摹ג ב-[9]



איור 2: תהליך עיבוד מלא: Complete Processing Pipeline

0.7 **תכונות מתקדמות נוספות:** Advanced Features

0.7.1 **טיפול במספרים מורכבים:** Complex Number Handling

הטבלה כוללת מספרים במגוון פורמטים:

- מספרים שלמים: 42, 1000000

- מספרים עשרוניים: 2.71828, 3.14159

- כתיב מדעי: 1.38e-23, 6.022e23

- אחוזים: 0.01%, 99.99%

- שנים: 1948, 2025

0.7.2 **שילוב תוכן מורכב:** Complex Content Integration

התבנית מאפשרת שילוב של:

1. טקסט דו-כיווני עם מעברים חלקים

2. קוד בשפות תכנות שונות

3. נוסחאות מתמטיות מורכבות

4. טבלאות עם תוכן מעורב

5. איורים ודיגרמות

6. ביבליוגרפיה דו-לשונית

0.8 סיכום ומסקנות מתקדמות: Advanced Summary and Conclusions

המסמך הדגים יכולות מתקדמות רבות:

- **ביבליוגרפיה:** ציטוטים מרובים ומורכבים עם הפניות לעמודים
- **טבלאות:** טבלאות מורכבות עם 6 עמודות ונתונים מגוונים
- **קוד:** דוגמאות מרובות עם PyTorch ועיבוד טקסט
- **מתמטיקה:** משוואות מרובות עם מספור והפניות צולבות
- **איורים:** דיאגרמות מורכבות עם TikZ
- **הפניות:** קישורים בין כל האלמנטים במסמך

התוצאות מראות שהתבנית מסוגלת לתמוך במסמכים אקדמיים מורכבים ביותר.

0.9 מקורות בעברית

- ד. כהן, ש. לוי, dna מ. אברהם, "עיבוד שפה טבנית בעברית: אתגרים ופתרונות," 3 כתוב עת לכלי לשונות חישוביות, 3, 51, .on, 234–256, 3202.
- מ. ישראלי dna ר. כהן, *בלשנות עברית מודרנית: תיאוריה ויישום*. ירושלים: הוצאת האוניברסיטה העברית, 2202, 512.
- ג. אברהם dna ל. שמעון, "אתגרים חישוביים בעיבוד טקסט עברי", 11 מחקרים מחשב ושפה, 2, .on, 112–128, 1202, .lov.

0.10 English References

- 1 A. Vaswani et al., "Attention is all you need," in *Advances in neural information processing systems*, 2017, 5998–6008.
- 2 J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- 5 J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *Proceedings of NAACL-HLT*, 4171–4186, 2019.
- 6 T. B. Brown et al., "Gpt-3: Language models are few-shot learners," OpenAI, Tech. Rep., 2020.
- 7 A. M. Turing, "Computing machinery and intelligence," in *Mind*, 59, 1950, 433–460.
- 8 C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, 379–423, 1948.
- 9 A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," in *OpenAI blog*, 1, 2019, 9.
- 10 T. Brown et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, 1877–1901, 2020.
- 12 W. Zhang, X. Chen, and Y. Liu, "A survey of natural language processing techniques," *ACM Computing Surveys*, vol. 54, no. 5, 1–36, 2022.
- 13 C. M. Bishop and H. Bishop, *Deep Learning: Foundations and Concepts*. New York: Springer, 2021.