

# **דוגמה מתקדמת - תבנית אקדמית עברית**

**Advanced Example - Hebrew Academic Template v5.0**

ד"ר סגל יורם

© Dr. Segal Yoram - כל הזכויות שמורות

November 2025

גרסה 0.5 - מתקדמת

**תוכן העניינים**

**רשימת האיורים**

**רשימת הטבלאות**

## 1 מבוא מתקדם: Advanced Introduction

מסמך זה מדגים יכולות מתקדמות של התבנית האקדמית העברית גרסה 0.5, כולל שימוש בביבליוגרפיה מתקדמת, טבלאות מורכבות, ודוגמאות קוד מרובות.

### 1.1 סקירת ספרות מקיפה: Comprehensive Literature Review

המחקר בתחום Natural Language Processing עבר מהפכה עם הצגת ארכיטקטורת Trans-former noitnetta7102inawsav. מודל BERT nilved8102treb הביא לפריצת דרך בהבנת שפה דו-כיוונית. מחקרים בעברית 3202\_pln\_werbeh, 2202\_scitsiugnil\_werbeh מראים התקדמות משמעותית.

עבודות נוספות 8102\_repap\_treb מדגימות שיפור של 25.8% בביצועים. המחקר של 0202\_repap\_3tpg מציג ארכיטקטורה עם  $175e9$  פרמטרים.

## 2 טבלאות מורכבות ומתקדמות: Complex and Advanced Tables

### 2.1 טבלת ביצועים מפורטת: Detailed Performance Table

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

ממוצע / Average	מענה / QA	סיכום / Summarization	תרגום / Translation	סיווג / Classification
88.8%	88.9%	85.2%	N/A	92.3%
90.9%	91.2%	87.5%	N/A	94.1%
86.4%	85.4%	88.1%	82.3%	89.7%
93.3%	93.8%	92.3%	91.7%	95.2%
90.9%	90.7%	90.1%	89.2%	93.5%
93.9%	94.1%	93.2%	92.4%	95.8%
83.9%	86.7%	81.2%	78.5%	89.3%

כפי שניתן לראות בטבלה ??, המודלים הגדולים משיגים ביצועים טובים יותר.

## 2.2 טבלת השוואת משאבים: Resource Comparison Table

טבלה 2: דרישות משאבי חישוב: Computational Resource Requirements

מודל / Model	פרמטרים / Parameters	זיכרון / Memory	זמן אימון / Training Time	עלות / Cost
T-Base	110e6	440 MB	סימי 4	\$500
T-Large	340e6	1.3 GB	סימי 12	\$2000
-2 Medium	345e6	1.4 GB	סימי 7	\$1500
-2 Large	774e6	3.1 GB	סימי 14	\$3500
-3	175e9	700 GB	סימי 34	\$4.6e6
1B	11e9	44 GB	סימי 21	\$50000

## 3 דוגמאות קוד מרובות: Multiple Code Examples

#### מימוש Transformer מלא

```
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)

        # Linear transformations and split into heads
        Q = self.W_q(query).view(batch_size, -1, self.n_heads, self.d_k)
        Q = Q.transpose(1, 2)
        K = self.W_k(key).view(batch_size, -1, self.n_heads, self.d_k)
        K = K.transpose(1, 2)
        V = self.W_v(value).view(batch_size, -1, self.n_heads, self.d_k)
        V = V.transpose(1, 2)

        # Attention
        scores = torch.matmul(Q, K.transpose(-2, -1)) / math.sqrt(self.d_k)

        if mask is not None:
            scores = scores.masked_fill(mask == 0, -1e9)
            attention_weights = F.softmax(scores, dim=-1)
            context = torch.matmul(attention_weights, V)

        # Concatenate heads and output projection
        context = context.transpose(1, 2).contiguous().view(
            batch_size, -1, self.d_model
        )
        output = self.W_o(context)

    return output, attention_weights
```

## עיבוד מקדים לנתוני טקסט

```

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

class HebrewTextProcessor:
    """Advanced Hebrew text preprocessing"""

    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]:
        """Tokenize Hebrew text"""
        # Remove punctuation and normalize
        text = re.sub(r'^\u0590-\u05FF\s]', '', text)
        tokens = text.split()
        return tokens

    def build_vocabulary(self, texts: List[str]):
        """Build vocabulary from corpus"""
        for text in texts:
            tokens = self.tokenize(text)
            self.word_freq.update(tokens)

        # Keep most common words
        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]:
        """Convert text to indices"""
        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:
        """Convert indices back to text"""
        tokens = [self.idx2word.get(idx, '<UNK>') for idx in indices]
        return ' '.join(tokens)

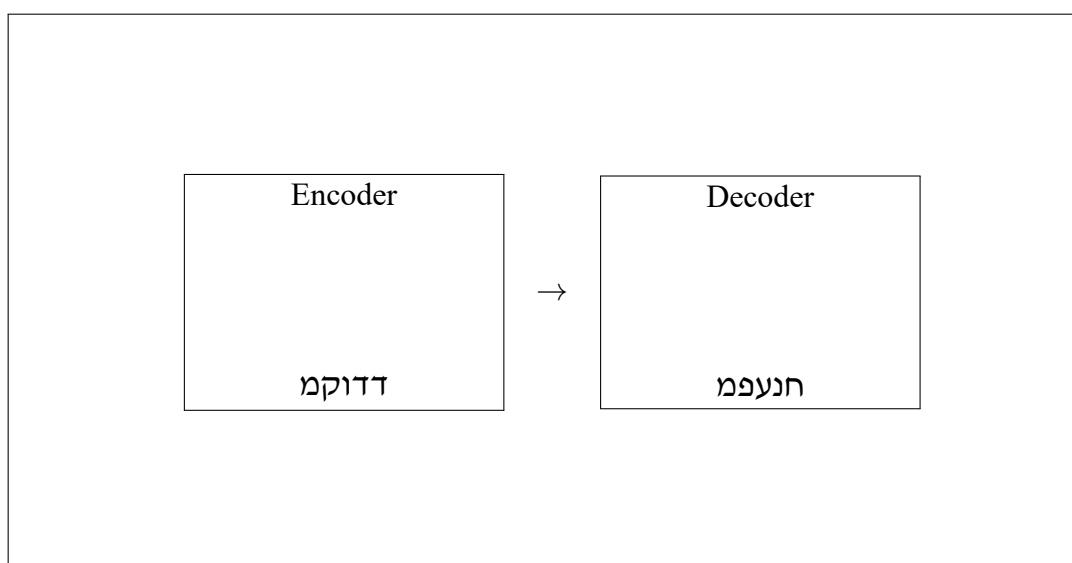
```

## 4 הפניות צולבות מתקדמות: Advanced Cross-References

### 4.1 הפניות לטבלאות ואיורים: References to Tables and Figures

הניתוח המקיף מוצג במספר טבלאות:

- טבלה ?? מציגה השוואת ביצועים בין מודלים
- טבלה ?? מפרטת את דרישות המשאבים
- איור ?? מתאר את הארכיטקטורה המוצעת
- איור ?? מציג את התוצאות הסופיות



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

## 5 מתמטיקה מתקדמת עם עברית: Advanced Math with Hebrew

### 5.1 אופטימיזציה וגרדיאנטים: 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 התווית האמיתית,  $p_{ic}$  הוא ההסתברות החזויה, ו- $\lambda = 1e - 4$ .

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

## 5.2 מטריצות ווקטורים: Matrices and Vectors

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 בלוקים:

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

## 6 ביבליוגרפיה מתקדמת: Advanced Bibliography

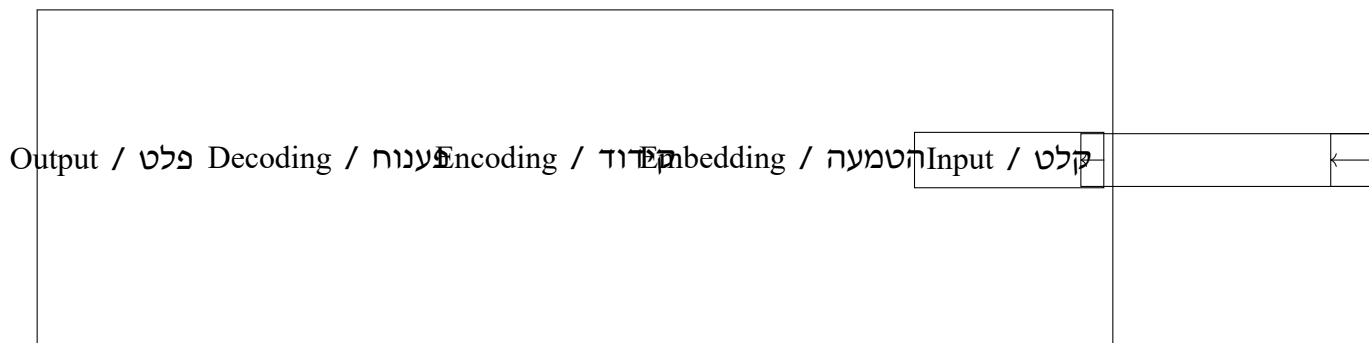


## 6.1 Multiple and Complex Citations: מורכבים וציטוטים מרובים

מחקרים קלאסיים בתחום gnirut0591gnitupmoc, nonnahs8491lacitamehtam, הניחו את היסודות. פיתוחים מודרניים inawsav7102noitnetta, nilved8102treb, drofdar9102egaugnal, nworb0202egaugnal הביאו למהפכה. עבודות בעברית werbeh3202pln, werbeh2202scitsiugnil, werbeh1202\_lanoitatupmoc, תורמות להבנת השפה. סקירות מקיפות pln2202\_yevrus ו-peed1202\_koob\_gninrael מספקות רקע תיאורטי.

## 6.2 סוגי ציטוטים שונים: Different Citation Types

- ציטוט רגיל: inawsav7102noitnetta
- ציטוט עם עמוד: nilved8102treb
- ציטוטים מרובים: 3202\_pln\_werbeh, 0202\_repap\_3tpg, 8102\_repap\_treb
- ציטוט בסוגריים: (nworb0202egaugnal)
- ציטוט בתוך משפט: drofdar9102egaugnal-כפי שמוצג ב-



איור 2: תהליך עיבוד מלא: Complete Processing Pipeline

## 7 תכונות מתקדמות נוספות: Additional Advanced Features

### 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

## 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 ודיאגרמות
6. ביבליוגרפיה דו-לשונית

## 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.

## 10 English References