

# מדריך למשתמש בשירות DSPy

## תוכן העניינים

3	1 חלק א': רקע והבנת הבסיס
3	1.1 1. מהם מודלי שפה גדולים (LLMs)?
3	1.2 2. מהו פרומפט (Prompt)?
3	1.3 3. הבעיה: כתיבת פרומפטים היא אמנות
4	1.4 4. מהו DSPy?
4	1.5 5. מהו אופטימיזר?
4	2 חלק ב': GEPA - האופטימיזר המתקדם
4	2.1 6. מהו GEPA?
5	2.2 7. איך GEPA עובד - הסבר מפורט
5	2.3 8. למה הרפלקציה כל כך חשובה?
6	3 חלק ג': פרמטרים של GEPA
6	3.1 9. פרמטרי תקציב (Budget Parameters)
6	3.2 10. פרמטרי מיזוג (Merge Parameters)
6	3.3 11. פרמטרי הערכה (Evaluation Parameters)
7	3.4 12. פרמטרי מודל הרפלקציה (Reflection LM)
7	4 חלק ד': מושגי יסוד בהגדרת משימה
7	4.1 13. Signature (חתימה)
7	4.2 14. Dataset (סט נתונים)
7	4.3 15. Metric (מדד) - דרישות מיוחדות ל-GEPA

<b>9</b>	<b>5 חלק ה': זרימת העבודה המלאה</b>
9	5.1 16. סקירת התהליך . . . . .
9	5.2 17. מבנה הבקשה לשרת . . . . .
10	5.3 18. רשימת נקודות קצה (API Endpoints) . . . . .
10	5.4 19. ניטור התקדמות . . . . .
<b>11</b>	<b>6 חלק ו': טיפים ושיטות עבודה מומלצות</b>
11	6.1 20. טיפים להכנת נתונים . . . . .
11	6.2 21. טיפים לכתיבת מדדים ל-GEPA . . . . .
11	6.3 22. פתרון בעיות נפוצות . . . . .
<b>12</b>	<b>7 חלק ז': מילון מונחים</b>

# 1 חלק א': רקע והבנת הבסיס

## 1.1 מהם מודלי שפה גדולים (LLMs)?

מודלי שפה גדולים (Large Language Models) הם תוכנות מחשב מתקדמות שעברו אימון על כמויות עצומות של טקסט מהאינטרנט, ספרים ומסמכים. דוגמאות מוכרות כוללות את ChatGPT, Gemini ו-Claude.

### מה הם יודעים לעשות:

- \* לענות על שאלות בשפה טבעית
- \* לכתוב טקסטים, סיכומים ותרגומים
- \* לנתח מידע ולהסיק מסקנות
- \* לפתור בעיות לוגיות ומתמטיות
- \* לעזור בכתיבת קוד

## 1.2 מהו פרומפט (Prompt)?

פרומפט הוא ההוראה או השאלה שאנחנו שולחים למודל השפה. איכות הפרומפט משפיעה באופן דרמטי על איכות התשובה.

### דוגמה לפרומפט פשוט: "סכם את הכתבה הבאה."

**דוגמה לפרומפט מפורט:** "אתה אנליסט פיננסי בכיר. סכם את הכתבה הבאה ב-3 נקודות מפתח, תוך התמקדות ברווחים הרבעוניים והתחזית לשנה הבאה. שמור על טון מקצועי ותמציתי."

הפרומפט השני ייתן תשובה ממוקדת יותר כי הוא מגדיר:

- \* את תפקיד המודל ("אנליסט פיננסי")
- \* את המיקוד ("רווחים ותחזית")
- \* את הפורמט והטון ("3 נקודות, מקצועי")

## 1.3 הבעיה: כתיבת פרומפטים היא אמנות

כתיבת פרומפטים אפקטיביים (Prompt Engineering) היא משימה מורכבת:

- \* **זמן:** נדרשים ניסויים רבים כדי למצוא את הניסוח האופטימלי.
- \* **מומחיות:** דורש הבנה של איך מודלים מפרשים הוראות.
- \* **עקביות:** קשה לשמור על ביצועים עקביים.

\* **סקיילביליות:** מה שעובד למשימה אחת לא בהכרח יעבוד לאחרת.

**הפתרון של DSPy:** כאן נכנס DSPy לתמונה: במקום לכתוב פרומפטים ידנית, אנחנו מגדירים את המשימה ונותנים לאלגוריתם אופטימיזציה לכתוב את הפרומפט הטוב ביותר עבורנו!

## 1.4 מהו DSPy?

DSPy (ראשי תיבות של Declarative Self-improving Python) היא תוכנה שפותחה באוניברסיטת סטנפורד ומאפשרת לתכנת מודלי שפה בצורה שיטתית.

העיקרון המרכזי: במקום לכתוב פרומפטים, אתה מגדיר:

1. **מה הקלט** - איזה מידע המודל מקבל
2. **מה הפלט** - מה אנחנו רוצים שהמודל יחזיר
3. **איך מודדים הצלחה** - מה נחשב לתשובה טובה

לאחר מכן, אלגוריתם אופטימיזציה לומד מדוגמאות ומייצר את הפרומפט האופטימלי אוטומטית.

## 1.5 מהו אופטימיזר?

אופטימיזר הוא אלגוריתם שמנסה למצוא את ההגדרות הטובות ביותר למשימה מסוימת. בהקשר של DSPy, האופטימיזר:

1. מקבל דוגמאות של קלט ופלט רצוי
2. מנסה וריאציות שונות של פרומפטים
3. מודד את ההצלחה של כל וריאציה
4. לומד מהניסויים ומשתפר
5. מחזיר את הגרסה הטובה ביותר

## 2 חלק ב': GEPA - האופטימיזר המתקדם

### 2.1 6. מהו GEPA?

GEPA (ראשי תיבות של Genetic-Pareto) הוא אופטימיזר מתקדם ב-DSPy שמשתמש ברפלקציה אבולוציונית לשיפור פרומפטים. מה שמייחד את GEPA הוא שבמקום להסתמך על תגמולים מספריים בלבד (כמו Reinforcement Learning), הוא מנתח את כל מסלול הביצוע (execution traces) - כולל נתיבי חשיבה, פלטי כלים, ואפילו שגיאות - ומשתמש בתובנות אלו לאבולוציה של פרומפטים טובים יותר.

## 2.2 7. איך GEPA עובד - הסבר מפורט

GEPA פועל בלולאה איטרטיבית של שיפור מתמיד (מחזור העבודה):

### 1. שלב 1: אתחול (Initialization)

- \* מתחיל עם פרומפט בסיסי (baseline).
- \* יוצר אוכלוסייה ראשונית של מועמדים (candidates).
- \* כל מועמד הוא וריאציה שונה של הפרומפט.

### 2. שלב 2: הערכה (Evaluation)

- \* כל מועמד מורץ על סט האימון.
- \* המערכת אוספת: ציונים מספריים (0-1), משוב טקסטואלי מפורט, מסלולי ביצוע (traces), ושגיאות.

### 3. שלב 3: רפלקציה (Reflection) - הלב של GEPA

- \* מודל הרפלקציה (reflection\_lm) מנתח את התוצאות.
- \* שואל: "למה פרומפט X הצליח ופרומפט Y נכשל?"
- \* מזהה דפוסים (אילו ניסוחים עובדים טוב יותר) ומפיק תובנות לשיפור.
- \* **זה מה שמבדיל את GEPA: הוא לא רק מודד - הוא מבין!**

### 4. שלב 4: אבולוציה (Evolution)

- \* ה-instruction\_proposer מציע פרומפטים חדשים המשלבים את התובנות מהרפלקציה.
- \* משתמש בטכניקות כמו מיזוג (merge) של פרומפטים מוצלחים, מוטציות ממוקדות על בסיס המשוב, ושילוב אלמנטים מהמועמדים הטובים ביותר.

### 5. שלב 5: סלקציה (Selection)

- \* בחירת המועמדים הטובים ביותר לדור הבא באמצעות עקרון פארטו (Pareto) לאיזון בין מטרות, תוך שמירה על גיוון באוכלוסייה.
- חזרה לשלב 2 עד להתכנסות או סיום התקציב.

## 2.3 8. למה הרפלקציה כל כך חשובה?

אופטימיזציה רגילה	GEPA עם רפלקציה
פרומפט A ← ציון 0.3 פרומפט B ← ציון 0.5 פרומפט C ← ציון 0.4	פרומפט A ← ציון 0.3 משוב: "נכשל בשאלות מרובות-שלבים" פרומפט B ← ציון 0.5 משוב: "הצליח כשהוסיף חשיבה צעד-אחר-צעד"
<b>מסקנה:</b> B טוב יותר. אבל למה? לא יודעים.	<b>רפלקציה:</b> "צריך להוסיף הוראה לחשיבה מובנית ובדיקה עצמית בסוף" → <b>דור הבא:</b> פרומפט משופר המשלב את התובנות.

### 3 חלק ג': פרמטרים של GEPA

#### 3.1 9. פרמטרי תקציב (Budget Parameters)

GEPA משתמש בגישת "תקציב" במקום פרמטרים מפורשים כמו מספר סיבובים.

פרמטר	תיאור
auto	הגדרת תקציב אוטומטית. ערכים אפשריים: □ <b>"light"</b> - תקציב קטן, ריצה מהירה (מתאים לניסויים ראשוניים). □ <b>"medium"</b> - תקציב בינוני (איזון בין מהירות לאיכות). □ <b>"heavy"</b> - תקציב מקסימלי, התוצאות הטובות ביותר ( <b>מומלץ!</b> ).
max_full_evals	חלופה ל-auto: מספר מקסימלי של הרצות מלאות (rollouts) שהאופטימיזר יכול לבצע. לשימוש כשרוצים שליטה מדויקת בתקציב.

#### 3.2 10. פרמטרי מיזוג (Merge Parameters)

פרמטר	תיאור
use_merge	האם להשתמש באופטימיזציה מבוססת מיזוג. ברירת מחדל: True.
max_merge_invocations	מספר מקסימלי של פעולות מיזוג. ברירת מחדל: 5.

#### 3.3 11. פרמטרי הערכה (Evaluation Parameters)

פרמטר	תיאור
num_threads	מספר threads להערכה מקבילית. ערך גבוה יותר = ריצה מהירה יותר (לדוגמה: 32).
failure_score	הציון שניתן לדוגמאות שנכשלו. ברירת מחדל: 0.0.
perfect_score	הציון המקסימלי האפשרי. ברירת מחדל: 1.0. (לזיהוי מתי הכל מושלם).
track_stats	האם לעקוב ולשמור סטטיסטיקות (לניתוח ודיבוג).

### 3.4 12. פרמטרי מודל הרפלקציה (Reflection LM)

**חשוב!** מודל הרפלקציה הוא קריטי להצלחת GEPA! מומלץ להשתמש במודל חזק עם יכולות חשיבה טובות.

\* **reflection\_lm**: מודל השפה לביצוע הרפלקציה.

## 4 חלק ד': מושגי יסוד בהגדרת משימה

### 4.1 13. Signature (חתימה)

החתימה היא הגדרה פורמלית של המשימה. היא מציינת:

\* **שדות קלט**: מה המודל מקבל (למשל: שאלה).

\* **שדות פלט**: מה המודל צריך להחזיר (למשל: תשובה).

\* **תיאור המשימה**: הסבר כללי על מה צריך לעשות.

### 4.2 14. Dataset (סט נתונים)

סט הנתונים הוא אוסף דוגמאות שמראות למודל מה אנחנו מצפים. **כמות מומלצת**: מינימום 30, מומלץ 50-100, אופטימלי +200.

**טיפ חשוב לחלוקת נתונים ב-DSPy**: בניגוד ללמידת מכונה קלאסית, ב-DSPy **מומלץ יותר Validation מאשר Train** עבור אופטימיזרי פרומפטים. הטווח המומלץ לכל סט הוא 30-300 דוגמאות.

### 4.3 15. Metric (מדד) - דרישות מיוחדות ל-GEPA

**קריטי!** המדד ב-GEPA שונה ממדדים רגילים. הוא חייב:

1. לקבל 5 פרמטרים: gold, pred, trace, pred\_name, pred\_trace.

מטרה	סט
ללמוד מהדוגמאות ולייצר פרומפטים	Train (אימון)
לבחור בין גרסאות שונות (מומלץ: יותר מ-Train!)	Validation (ולידציה)
למדוד ביצועים סופיים על נתונים שלא נראו	Test (מבחן)

2. להחזיר dspy.Prediction עם:

\* **score**: ציון מספרי בין 0 ל-1.

\* **feedback**: משוב טקסטואלי מפורט.

**דוגמה ללוגיקת מדד:**

\* **תשובה זהה לחלוטין**: ציון 1.0, משוב "תשובה מושלמת!".

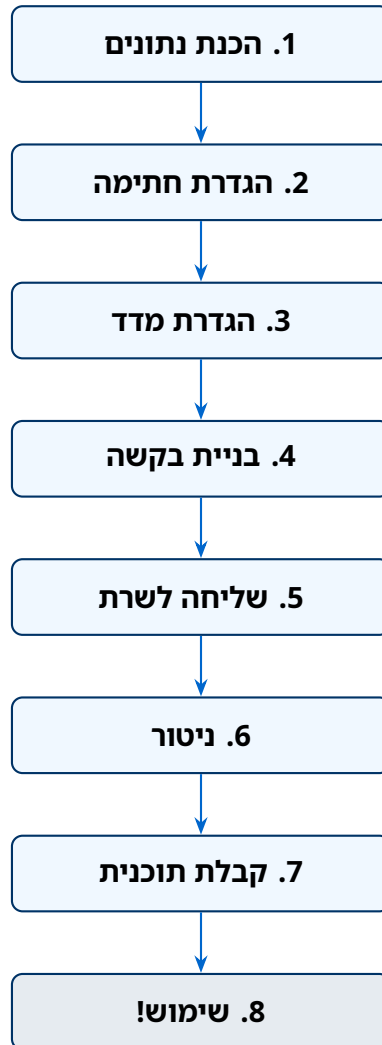
\* **תשובה חלקית**: ציון 0.5, משוב "תשובה חלקית. הנכון הוא X, קיבלנו Y. חסר Z".

\* **תשובה שגויה**: ציון 0.0, משוב "שגוי לחלוטין. המודל התבלבל בין A ל-B...".



## 5 חלק ה': זרימת העבודה המלאה

### 5.1 16. סקירת התהליך



### 5.2 17. מבנה הבקשה לשרת

שדות חובה ואופציונליים בבקשה:

שדה	תיאור
module_name	סוג המודול. לרוב "dspy.ChainOfThought". (מוסיף שדה reasoning לפני הפלט, משפר ביצועים במשימות מורכבות).
optimizer_name	שם האופטימיזר ("dspy.GEPA").
signature_code	קוד החתימה (כטקסט).
metric_code	קוד המדד (כטקסט).

שדה	תיאור
dataset	רשימת הדוגמאות (JSON).
column_mapping	מיפוי בין שמות השדות בחתימה לבין שמות העמודות בנתונים (למשל: אם בחתימה יש question ובנתונים q).
model_config	הגדרות מודל השפה הראשי.
reflection_lm	הגדרות מודל הרפלקציה ( <b>חובה ל-GEPA!</b> ).
optimizer_kwargs	פרמטרים לאופטימיזר (כמו "auto=heavy").
split_fractions	(אופציונלי) יחסי החלוקה ל-Train/Val/Test (למשל [0.2, 0.3, 0.5]).
shuffle	(אופציונלי) האם לערבב את הנתונים לפני החלוקה.
seed	(אופציונלי) זרע אקראיות לשחזור תוצאות (Reproducibility).

### 5.3 18. רשימת נקודות קצה (API Endpoints)

להלן הכתובות המרכזיות לעבודה מול השירות:

מטרה	כתובת (Endpoint)
שליחת עבודת אופטימיזציה חדשה.	/run
בדיקת סטטוס כללי של העבודה.	/jobs/{id}
קבלת סיכום תוצאות, ציונים וגרפים.	/jobs/{id}/summary
צפייה בלוגים של הריצה לדיבוג.	/jobs/{id}/logs
הורדת הקובץ המאופטם הסופי.	/jobs/{id}/artifact
בדיקת תקינות השרת.	/health

### 5.4 19. ניטור התקדמות

- \* **pending**: ממתין בתור.
- \* **running**: האופטימיזציה רצה כעת.
- \* **completed**: הסתיים בהצלחה.
- \* **failed**: נכשל (בדוק לוגים).

## 6 חלק ו': טיפים ושיטות עבודה מומלצות

### 6.1 20. טיפים להכנת נתונים

- \* איכות מעל כמות (50 מדויקות עדיפות על 200 בינוניות).
- \* גיוון (מקרי קצה).
- \* ניקיון (ללא שגיאות).
- \* עקביות וייצוגיות.

### 6.2 21. טיפים לכתיבת מדדים ל-GEPA

- \* **משוב מפורט:** הסבירו בדיוק מה היה לא נכון.
- \* **משוב קונסטרוקטיבי:** ציינו מה צריך לשפר.
- \* **ציון הוגן:** תנו ניקוד חלקי.
- \* **עקביות:** ציונים דומים למקרים דומים.

### 6.3 22. פתרון בעיות נפוצות

בעיה	פתרון
העבודה נכשלת מיד	בדקו פורמט בקשה ושגיאות JSON.
תוצאות גרועות	הוסיפו דוגמאות מגוונות, שפרו את המשוב במדד.
ריצה איטית מאוד	השתמשו ב- <code>auto="light"</code> לניסוי ראשוני, הגדילו <code>num_threads</code> .
המדד תמיד נותן 0	בדקו את לוגיקת ההשוואה, הוסיפו גמישות (אותיות קטנות, רווחים).
הרפלקציה לא עוזרת	שדרגו את מודל הרפלקציה, שפרו את המשוב במדד.

## 7 חלק ז': מילון מונחים

מונח	הסבר
LLM	מודל שפה גדול.
Prompt	הוראה שנשלחת למודל.
DSPy	תוכנה לתכנות מודלי שפה (Declarative Self-improving Python).
Optimizer	אלגוריתם שמשפר פרומפטים אוטומטית.
GEPA	Genetic-Pareto - אופטימיזר עם רפלקציה.
Reflection	תהליך הניתוח של הצלחות/כישלונות.
Signature	הגדרת קלט/פלט של המשימה.
Metric	פונקציה לבדיקת הצלחה ומתן משוב.
Dataset	אוסף דוגמאות לאימון והערכה.
Train / Val / Test	סטים ללמידה, בחירה (ולידציה) ומדידה סופית.
Job	עבודת אופטימיזציה בשרת.
Artifact	התוכנית המאופטמת הסופית.
Rollout	הרצה מלאה אחת של התוכנית.
Trace	מסלול הביצוע (כל השלבים שהמודל עבר).
ChainOfThought	מודול שמוסיף חשיבה צעד-אחר-צעד.
reflection_lm	מודל השפה שמבצע את הרפלקציה ב-GEPA.