הפדגוגיה של הבינה המלאכותית Stacks

כתבו:
רחל פרלמן, הוראת מדעי המחשב והנדסת תוכנה
אורית חזן, הפקולטה לחינוך למדע וטכנולוגיה, הטכניון
בשיתוף יחד עם מורי ומורות מדעי המחשב המלמדים עם Stacks

מסמך זה מתאר את הפדגוגיה שעומדת בבסיס פעילותו של הבוט Stacks במקרה של הוראת מדעי המחשב. בשנת הלימודים תשפ"ו, 50 בתי ספר לומדים מדעי המחשב בשילוב Stacks. בנוסף לתיאור להלן לפדגוגיה עליה מבוסס הבוט, מורי ומורות מדעי המחשב המלמדים כעת יחד עם הבוט מפתחים שיטות חדשות להוראת מדעי המחשב. שיטות אלה נלמדות כעת ותתווספנה למסמך בהמשך. בנוסף להסבר כללי של הפדגוגיה, מוצגת דוגמא של פתרון בעיות של תלמיד המונחה ע"י Stacks כולל ניתוח פדגוגי של הדיאלוג.

הפדגוגיה של הבינה המלאכותית המהווה את בסיס פעילותו של Stacks

הפדגוגיה של הבינה המלאכותית ב-Stacks מבוססת על העקרונות הבאים:

I. **למידה פעילה (active learning) ומותאמת אישית (personalized learning):** המורה "בונה" את סביבת הלימוד כמעין פאזל של תרגילים אינטראקטיביים ומותאמים לרמת הכתה באופן כללי, יכולות ה-AI מסייעות ליצור ולהתאים תכנים לכל תלמיד ותלמידה בהתאם לרמת וצרכי התלמידים, בהתאם לאינטראקציה שהתלמיד/ה עם הבוט. במילים אחרות, התלמידים עובדים בסביבה אינטראקטיבית שבה הבינה המלאכותית משמשת עוזרת הוראה אישית. בעת כתיבת הקוד, הבוט מזהה את הקושי, שואל שאלות מנחות, ומכוון את הלומד צעד-אחר-צעד עד להשלמת הפתרון.

II. **משוב רציף ברוח הקונסטרקטיניזם (constructionism):** הנחת העבודה היא כי בתהליך הפיתוח, התלמידים בונים מבנים מנטלים של מושגים במדעי המחשב כאשר המשוב שנותן הבוט מנחה אותן בתהליך הבניה. זהו תהליך קונסטרקטיניזם היות וכל משוב מותאם אישי לתהליך בניית המבנים בראש התלמידים במקביל בניית עצמים - תוכניות מחשב. מענה זה הוא דיפרנציאלי: התאמת המשוב לרמת התלמידים, כולל רמזים, ודוגמאות. תכונה זו חשובה במיוחד בכתות הטרוגניות.

III. **שיפור מיומנויות:** בנוסף לבניית המבנים המנטלים בראש התלמידים, המשוב הרציף שנותן Stacks מכוון לשיפור מיומנויות פתרון בעיות וחשיבה חישובית, כולל היבטים חברתיים וניהוליים. בין השאר, במערכת משלבות בדיקות אוטומטיות (בדיקות קלט-פלט ובדיקות פונקציונליות), כך שהתלמידים מקבלים היזון חוזר מיידי תוך התנסות אמיתית בכתיבת קוד, ובו בזמן, למורים יש בקרה פדגוגית מלאה על התהליך—מהגדרת מטרות ההוראה והלמידה ועד בניית תרגילים המקדמים חשיבה ביקורתית, מוטיבציה ורפלקציה.

IV. **הסרת חסמים וכניסה פשוטה:** הסביבה היא וובית ואפשר להתחיל לעבוד בה באופן אינטואיטיבי.

תפקיד המורה ב-Stacks (ובכיתה עם AI בכלל)

תפקיד המורה הוא כ"מנצח פדגוגי" המגדיר מסלול, מפקח על האיכות, ומתרגם נתונים ללמידה משמעותית. באופן כללי, הבינה המלאכותית (Stacks) מטפלת באוטומציה ובבדיקות, אך האחריות הפדגוגית היא של

המורים, כולל וידוי שהטכנולוגיה מקדמת למידה אמיתית. הנה עיקר הפעילויות הפדגוגיות של המורים בתהליך ההוראה עם Stacks:

1. הגדרת מטרות הוראה ולמידה וקריטריונים להערכה;
2. עיצוב משימות בשלבים עם הדרגתיות, לטיפוח חשיבה אלגוריתמית ופתרון בעיות;
3. שימוש בנתונים שמספקת המערכת לזיהוי פערים, ביצוע התאמות בזמן אמת, ומתן משוב מעצים (ולא רק ציון סופי);
4. שמירה על אתיקה ואמינות אקדמית: ניסוח כללי שימוש בבינה מלאכותית, ציון מקורות התרגילים;
5. הערכה מבוססת גם על תהליך העבודה (ולא רק המוצר הסופי), הכוללת הסבר מפורט על תהליך הלמידה;
6. הוראת אוריינות-AI: כיצד לייצר פרומפטים איכותיים, לאמת תוצרים, ולהבין את מגבלות הבינה המלאכותית ואבחנה בהטיות;
7. טפוח מיומנויות חברתיות-קוגניטיביות: שיתוף פעולה, רפלקציה, וחשיבה ביקורתית על מענה הבינה;
8. פיתוח מקצועי בהוראה ועדכון מקצוע ההוראה למנטור.

דוגמא: פתרון בעיות מבוסס דיאלוג מנחה למידה של תלמיד-בינה

כתובת IP מורכבת מ־4 חלקים (אוקטטים) המופרדים ע"י נקודה. כל חלק (אוקטטה) הינו מספר שלם בטווח 0–255 (כולל).

לדוגמה:
192.168.0.17 כתובת IP חוקית
192.890.0.17 כתובת IP לא חוקית
192.890.0.17.89 כתובת IP לא חוקית
192.890 כתובת IP לא חוקית

עליכם לכתוב פעולה המקבלת מחרוזת ובודקת האם כתובת IP המורכבת מספרות ומופרדת ע"י נקודה חוקית. על הפעולה להחזיר True אם הכתובת שהתקבלה חוקית, אחרת False.
יש לזמן את הפעולה פעמיים: פעם אחת עם כתובת חוקית ופעם נוספת עם כתובת לא חוקית.

הנה דיאלוג של תלמיד אחד עם Stacks בתהליך פתרון הבעיה עד להגעה לפתרון מלא.

מס' קטע	דיאלוג תלמיד-בינה	ניתוח פדגוגי
1	תלמיד: איך להתחיל את התרגיל? Stacks: "היי …, כדי להתחיל, אתם צריכים להגדיר את הפונקציה ip_octets כך שתקבל מחרוזת	פירוק בעיה מודרך: הבוט לא פותר, הוא מפרק את המשימה לתת-יעדים ברורים (הגדרת פונקציה, בדיקת 4 חלקים, בדיקת טווח 0–255).

	(string). ראשית, בדוק.י שהכתובת מורכבת בדיוק מארבעה חלקים המופרדים בנקודה. לאחר מכן ודא.י שכל חלק הוא מספר בטווח 0–255. שימו לב: אם אחד החלקים אינו מספר שלם, או שיש יותר/פחות מארבעה חלקים – הפונקציה צריכה להחזיר False. נסו להריץ עם כתובת חוקית ואז עם כתובת לא חוקית כדי לבדוק את עצמכם.	נורמות וקריטריונים גלויים: מציג תנאים מדויקים להצלחה (מבנה הכתובת, טווחים, טיפול בשגיאות) זהו שילוב של פירוק בעיה, רמזים מדורגים ומשוב מתמשך שמפתח הבנה עקרונית, לא רק פתרון נקודתי.

2	תלמיד: "למה הקוד שלי לא עובד?"
	קוד התלמיד:
	`def ip_octets(ip):`
	`  for oct in ip:`
	`  if oct > 0 and oct <= 255:`
	`  return True`
	`  else:`
	`  return False`
	
	`ip_octets("192.168.0.17")`
	`ip_octets("192.444.0.17")`
	
	Stacks: היי, הקוד שלך לא עובד כרגע מכיוון שאת.ה מנסה לעבור על החתיכות
	כאילו הן כבר רשימה של מספרים, אבל בפועל הן תווי מחרוזת.
	כדי לבדוק תקינות של כתובת IP את.ה צריכ.ה קודם לפצל את המחרוזת
	לארבעה חלקים לפי נקודה (.), ואז להמיר כל חלק למספר שלם.
	לאחר מכן בדוק.י שכל חלק בטווח 0–255 ושמספר החלקים הוא בדיוק ארבעה.
	רוצה שאנחה אותך שלב-שלב ואעזור לך לפצל תחילה לפי נקודה?
		הבינה המלאכותית מסבירה לא רק "מה לא בסדר", אלא גם למה—הסיבה הלוגית לשגיאה—ומציעה רצף הנחיות תיקון מדורג:
		פיצול המחרוזת לארבעה חלקים לפי נקודה, המרת כל חלק למספר שלם, בדיקה שכל חלק בטווח 0–255, אימות שמספר החלקים הוא בדיוק ארבעה, הרצה עם כתובת חוקית ולא-חוקית כדי לוודא שהפונקציה פועלת כמצופה.
		– משוב חכם בזמן אמת: הבינה המלאכותית מזהה שגיאות בקוד בזמן אמת ומספקת לתלמיד הסבר ממוקד ומתואם אישית. ה-AI לא רק מצביע על השגיאה, אלא גם מסביר את הסיבה הלוגית ומדריך את הלומד כיצד לתקן צעד-אחר-צעד.

3	התלמיד: כיצד לשפר את הקוד?
	קוד התלמיד:
	`def ip_octets(ip):`
	`  for oct in ip:`
	`  if oct > 0 and oct <= 255:`
	`  return True`
	`  else:`
	`  return False`
	
	`ip_octets("192.168.0.17")`
	`ip_octets("192.444.0.17")`
		הדיאלוג מתמשך, הבינה המלאכותית לא רק מציינת ומסמנת טעויות, אלא מציעה רמזים מדורגים ושאלות מכוונות כדי להוביל לפתרון תקין.

	Stacks: היי, את/ה שואל/ת איך לפרק מחרוזת ומה ההוראות. המטרה היא לפצל את כתובת ה-IP לחלקים (האוקטטים) לפי הנקודה (.). בינתיים, הנה דוגמה כללית לפיצול מחרוזת לערכים לפי מפריד: לדוגמה, אם יש לך "apple,banana,orange" ורוצים לפצל לפי פסיק, משתמשים ב-split(','). האם תרצה/י לנסות להשתמש ב-split('.') כדי לפצל את כתובת ה-IP לארבעה חלקים? אחרי הפיצול, נמיר כל חלק למספר שלם ונבדוק שכל אחד בטווח 0–255 ושיש בדיוק ארבעה חלקים. איך היית/היית מיישם/ת את הצעד הראשון – פיצול לפי נקודה?	

4	קוד התלמיד:
	`def ip_octets(ip):`
	`  lst = ip.split('.')  # פיצול הכתובת לארבעה חלקים (מחרוזות)`
	`  for oct in lst:`
	`  if oct >= 0 and oct <= 255:`
	`  return True`
	`  else:`
	`  return False`
	
	`ip_octets("192.168.0.17")`
	`ip_octets("192.444.0.17")`
	
	Stacks: היי, אני רואה שהשתמשת בפעולה split('.') — מצוין, כך פיצלת את הכתובת לחלקים.
	הבעיה היא שכעת אתה משווה מחרוזות למספרים בתנאי if, ולכן הבדיקה לא תקפה
	(למשל "192", "168", "0" הן טקסטים, לא מספרים).
	כדי להמשיך נכון:
	המר כל חלק למספר שלם בעזרת int(...).
	ודא שמספר החלקים הוא בדיוק 4.
	בדוק שכל חלק בטווח 0–255.
	איך תהפוך מחרוזת שמייצגת מספר למספר שלם בפייתון לפני בדיקת הטווח? (שאלה מכוונת)?
		דיוק תשובת הלומד: שוב, הבינה אינה רק "מסמנת טעות", אלא מסבירה מגדוע היא מתרחשת (השוואת מחרוזות למספרים), מציעה צעדי תיקון מדורג (המרה ל־int, ספירת חלקים, בדיקת טווח), ושואלת שאלה מנחה כדי לעודד את הלומד להשלים את התיקון בעצמו.

5	קוד התלמיד:
	`def ip_octets(ip):`
	`  lst = ip.split('.')  # מפצל את הכתובת לחלקים`
	`  for oct in lst:`
		ה-AI מזהה שהלומד כבר תיקן שתי נקודות קודמות (פיצול למחרוזות והמרה ל-int, בדיקת טווח).

	oct = int(oct) המרה למספר שלם # if oct < 0 or oct > 255: return False return True ip_octets("192.168.0.17") ip_octets("192.444.0.17") Stacks: היי, אני רואה שפיצלת את המחרוזת בעזרת split('.') וגם המרת כל חלק למספר — מצוין. עכשיו, לגבי הבדיקות: הקוד שלך בודק טווחים (0–255), אבל חשוב לוודא גם שמספר החלקים הוא בדיוק 4 אוקטטים. לדוגמה, כתובת כמו 192.168.0.17.89 או כזו עם פחות מ-4 חלקים אינה תקינה. הצעה: לפני לולאת הבדיקה, בדוק/י len(lst) == 4. שאלה מכוונת: איפה נכון להוסיף בדיקה שמוודאת שיש בדיוק 4 חלקים בכתובת ה-IP?	היא מוסיפה את מקרי הקצה המבניים: חובה לוודא מספר חלקים תקין. השיח נשאר דיאלוגי: במקום לתת קוד מלא, ה-AI שואלת שאלה מנחה ("איפה למקם את הבדיקה?") כדי לעודד הבנה ותיקון עצמאי.

6	קוד תלמיד:
	def ip_octets(ip):
	lst = ip.split('.') מפצל את הכתובת לחלקים #
	if len(lst) != 4: # בדיקת מבנה: בדיוק 4 חלקים
	return False
	for oct in lst:
	oct = int(oct) המרה למספר שלם #
	if oct < 0 or oct > 255:
	return False
	return True
	
	ip_octets("192.168.0.17")
	ip_octets("192.444.0.17")
	תשובת הבינה המלאכותית:
	היי, רואה שהוספת את הבדיקה if len(lst) != 4: return False — רעיון נכון!
	הקוד כבר מטפל בשניים משלושת השלבים:
	(1) פיצול הכתובת לארבעה חלקים, (2) בדיקת טווח לכל חלק (0–255).
	עכשיו יש עוד נקודת קצה שחשוב לשים לב אליה: מה קורה אם אחד החלקים אינו מספר?
	לדוגמה, כתובת כמו "192.168,.17" או "a.b.c.0" תעבור פיצול, אבל המרה ל־int
		ה-AI מזהה שהלומד כבר הוסיף בדיקת מבנה נכונה (בדיוק 4 חלקים) וטיפול בטווחים.
		היא מציפה טעות נפוצה נוספת: חלק שאינו מספר (Non-numeric), ומסבירה מדוע יש לתקן זאת. .
		במקום לתת פתרון מלא, ה-AI מציעה רמזים ושואלת שאלות מנחות (בדיקת ספרות או try/except) המעודדות את הלומד להשלים בעצמו את התיקון.

	תיכשל. איך תוודא/י שהקוד מטפל גם במקרה הזה, מבלי לגרום לשגיאה? רמז: אפשר לבדוק שהמחרוזת מייצגת ספרות בלבד לפני ההמרה, או ללכוד חריגה בזמן ההמרה. שאלה מכוונת: היכן נכון למקם את הבדיקה/הלכידה כדי לשמור על הקוד קריא ובטוח?	

7	קוד תלמיד:
	`def ip_octets(ip):`
	`  lst = ip.split('.')`
	`  if len(lst) != 4:`
	`  return False`
	`  for oct in lst:`
	`  oct = int(oct)`
	`  if not oct.isdigit():`
	`  return False`
	`  if oct < 0 or oct > 255:`
	`  return False`
	`  return True`
	
	`ip_octets("192.168.0.17")`
	`ip_octets("192.444.0.17")`
	
	Stacks: היי, אני רואה שאת/ה מנסה להשתמש ב-isdigit() אחרי שכבר המרת את oct למספר שלם.
	יש לשנות את סדר הפעולות: קודם לבדוק שהמחרוזת מייצגת ספרות בלבד בעזרת isdigit(), ורק לאחר מכן להמיר ל-int. אחרת תתקבל שגיאה/בדיקה לא תקפה כי לאובייקט מספרי אין isdigit().
		ה-AI מאבחנת את מקור השגיאה הלוגית (שימוש ב-isdigit() אחרי המרה ל-int).
		היא מסבירה מדוע זו טעות (ל-int אין isdigit()), ובכך מקדמת הבנה ואינה רק מתקנת טעות. .
		היא מציעה תיקון באופן הדרגתי עם סדר פעולות נכון (בדיקת מחרוזת → המרה → בדיקת טווח).
		לסיום, ה-AI שואלת שאלה מנחה/מציעה הדגמה במטרה לעודד את התלמיד להשלים את התיקון בעצמו.

8	קוד תלמיד:
	`def ip_octets(ip):`
	`  lst = ip.split('.')`
	`  if len(lst) != 4:`
	`  return False`
	`  for oct in lst:`
	`  if not oct.isdigit():`
	`  return False`
	`  oct = int(oct)`
	`  if oct < 0 or oct > 255:`
	`  return False`
	`  return True`
		בקשת ציון מהבינה וקבלת משוב: משוב זה יוצר אצל הלומד בעלות על התהליך, שמחה בהישג, ורצון להתקדם הלאה. זה לא רק ציון סופי אלא חלוקת ציון ל-3 היבטים: תכנון, לוגיקה ואיכות קוד - מה בוצע טוב ומה יש לשפר, כך שהתלמיד מקבל הן מוטיבציה והן כלים לשפר ולדייק את תשובתו.

	print(ip_octets("192.168.0.17")) print(ip_octets("192.444.0.17")) Stacks, מתן ציון: היי, כל הכבוד על הפתרון שלך! תכנון: 10 / 10 לוגיקה: 10 / 10 איכות קוד: 10 / 10 מה עשית טוב: הקוד שלך מבצע בדיוק את מה שנדרש בצורה מדויקת ויעילה. הלוגיקה לבדיקת תקינות כתובת ה-IP נכונה ומטפלת בכל המקרים הנדרשים: מספר החלקים בכתובת, בדיקה שכל חלק מורכב מספרות בלבד, ובדיקה שכל חלק נמצא בטווח שבין 0–255. השתמשת בשמות משתנים ברורים המקלים על הבנת הקוד. זמנת את הפעולה פעמיים, פעם אחת עם כתובת חוקית ופעם עם כתובת לא חוקית, בדיוק כפי שהתבקש בתרגיל. זה מראה הבנה מעמיקה של החומר ויכולת תכנון קוד איכותי ומקצועי. המשיכ/י כך!	

סיכום התהליך הדיאלוג עם ה-AI בדוגמא לעיל של בדיקת כתובת IP

1. אבחון מיידי ומין-דיאגנוסטי: ה-AI מזהה את סוג הטעות (למשל: השוואת מחרוזת למספר, סדר פעולות שגוי, היעדר בדיקת len==4) ומדייק את מקור הבעיה—not רק "לא עובד".
2. רמזים מדורגים (Scaffolding): במקום לפתור במקומו, ה-AI נותן רמז → שאלה מכוונת → כלל פעולה. העזרה מתמעטת כשהלומד מתקדם.
3. הוראה לשם הבנה, לא רק תוצאה: הדגש הוא "למה זה שגוי" ו"מה העיקרון הכללי" (בדיקת תקינות קלט, טיפוסי נתונים, מקרי קצה), כדי להעביר ידע שמכליל למשימות עתידיות.
4. למידה מבוססת תהליך ומשוב רציף: כל תיקון מפיק משוב נוסף—מחזורי ניסוי-בדיקה-שיפור (iterate), עד שהלומד מגיע לפתרון תקין ומבין את הדרך.
5. שאלות סוקרטיות שמקדמות מטא-קוגניציה: שאלות כמו "איפה נכון למקם את הבדיקה?" גורמות ללומד להצדיק בחירות, להסביר קוד, ולפתח הרגלי חשיבה של מתכנת.
6. שליטה במקרי קיצון: ה-AI מוסיף בהדרגה בדיקות מבניות/לוגיות (4 חלקים בדיוק, isdigit לפני int, תחום 0–255), כך שהפתרון נהיה עמיד ו"יציב קצה".

7. בניית תחושת מסוגלות ומוטיבציה: המשוב המסכם מדגיש הישגים, מתרגם התקדמות לנקודות חוזק, ומייצר חוויית "אני מצליח/ה"—מניע להמשך תרגול.
8. אחריותיות ושקיפות: הקריטריונים גלויים (תכנון, לוגיקה, איכות קוד), והציון מלווה בנימוק—מה השתפר, מה עדיין לשפר.
9. התאמה אישית בזמן אמת: העזרה מותאמת למצב הנוכחי של הלומד: אם פוצל מחרוזת—נעבור להמרה; אם הומר—נעבור לאימות ספרות; וכו'.
10. העברת הבעלות ללומד: בסיום, הלומד הוא זה שמריץ, בודק, ומאמת מול מקרי מבחן—ה-AI רק מנווט. כך נבנית עצמאות מקצועית.

סיכום

ניתן לראות כי באה כאן לידי ביטוי פדגוגיה קונסטרקטיביסטית, המשולבת בתהליך הערכה מעצבת, בו ניתנים רמזים מדורגים ומוצגות שאלות מכוונות, המפתח הבנה עמוקה ומוטיבציה פנימית להתנסות ולהשתפר.