# Development of a Benchmark (Of NN algorithms) Corpus to Support Entity Recognition in Job Descriptions.

## תקציר

זיהוי וחילוץ של ישויות בולטות הוא משימה חשובה ביישומי חילוץ מידע רבים בעולם האמיתי כגון סיווג טקסט, אלגוריתמי חיפוש והמלצות תוכן.
גיוס עובדים, חברות כיום נהנות ממערכות אוטומטיות לרכוש מידע עדכני אודות תפקידי עבודה ופרופילי מועמדים מפורטים מבחינת מיומנויות, כישורים וניסיון.
עם זאת, הפיתוח של זיהוי ישויות (ER) לביצוע משימות אלה סובלים קשות בגלל היעדר datasets זמינים לציבור.
רוב ה datasets הזמינים מורכבים ממאמרי חדשות כללים(עיתונים וכו׳).

## מבוא

הזיהוי והחילוץ של ישויות בולטות הוא משימה חשובה ביישומי חילוץ מידע רבים בעולם האמיתי.
מחפשי עבודה וגיוס עובדים כאחד נהנות ממערכות אוטומטיות אשר יודעות לסווג תפקידים ופרופילי מועמדים.
עם זאת, הפיתוח של זיהוי ישויות (ER) לקויות בחסר בגלל העדר נתוני הכשרה הזמינים לציבור.
רוב המנגנוני ה ER מורכבים ממאמרי חדשות כלליים.

כדי לפתח כלים טובים יותר להתאמת עבודה, עלינו לטפל בבעיות הללו.
ראשית אנו קובעים הגדרה של הגורמים הרלוונטים.
שהם: skills,education,experience.
השתמשנו בדאטה סט הזה: train_rav_1 שהוא מציג תיאורי עבודה בבריטניה, שהוא זמין לציבור
בונים את הדאטה סט שלנו מעל הדאטה סט שהוגדר למעלה על פי שלושת הלייבלים:
skills,education,experience.
בתחילה, הישתמשנו בכלים לחילוץ ישויות מטקסט, כדי לסווג עצמאית את שלושת הלייבלים האלו.
לאחר מכן, אנו מציגים benchmark בין מודל x ומודל y.

בנוסף, הדאטה סט שנוצר הוא זמין לציבור, ואפשר להשתמש בו למודלים אחרים.
ניתן למצוא את קוד המקור ב- (קישור לקוד מקור).

## עבודות קודמות

ב-גישות למידת מכונה מסורתיות, כרוכה בניתוח טקסט לא מובנה וחילוץ פרטים בולטים ממנו.
ההנחה הבסיסית הוא כי דמיון גבוה בין סט הכישורים של מועמד ומערך הכישורים הנדרשים לעבודה הוא אינדיקטור חזק להתאמה טובה.
עם זאת, יש שתי סוגיות עיקריות בספרות הקיימת לגבי מיומנות חילוץ לבחירת תכונה.
ראשית, אין קונצנזוס אקדמי על הגדרת מיומנות, אשר עושה השוואה בין שיטות מיצוי שונות.

שנית, זה לא ברור, אילו סוגי שיטות יתנו ביצועים יותר טובים לחילוץ מיומנויות מטקסט בהינתן הלייבלים האלו.
לכן, יש צורך במחקר על הערכה של שיטות שונות.
לגבי הסוגייה הראשונה, במחקר הזה,
הכותבים לקחו dataset "מוכן" מ Kaggle 5(תפקידים בבריטניה), עיבדו אותו, ועבדו צמוד עם מחלקת משאבי אנוש כדי להבין איזה לייבלים הם הכי רלוונטים בכדי למצוא התאמה הכי טובה בין תיאור תפקיד לבין מועמדים, והם הגיעו למסקנה הזאת שאלו הם הלייבלים: Skills, Qualifications, Experiences, Domains, and Occupations.
כדי להבין שה dataset שלהם תואם למציאות, הם שלחו את dataset הזה לעובדי אמזון (200 עובדים).
על פי המאמר, 170 איש ענו נכון, וחלק "רפרפו" כי מיהרו.
בשביל לבדוק את התשובות של האנשים, החוקרים בנו מסווג שיודע לזהות האם יש טעויות באופן שיטתי בתשובות.
כל הישויות סומנו באמצעות ערכת IOB.
במאמר זה, החוקרים השתמשו באלגוריתם CRF, שהניב את התוצאות הבאות:

| Label | P | R | $F_1$ | Support |
|---|---|---|---|---|
| B-Skill | 0.69 | 0.37 | 0.48 | 676 |
| I-Skill | 0.53 | 0.71 | 0.61 | 1429 |
| B-Qualification | 0.72 | 0.50 | 0.59 | 26 |
| I-Qualification | 0.39 | 0.23 | 0.29 | 40 |
| B-Occupation | 0.90 | 0.65 | 0.75 | 137 |
| I-Occupation | 0.93 | 0.71 | 0.81 | 164 |
| B-Experience | 0.86 | 0.67 | 0.75 | 9 |
| I-Experience | 0.42 | 0.76 | 0.54 | 17 |
| B-Domain | 0.53 | 0.40 | 0.46 | 60 |
| I-Domain | 0.34 | 0.28 | 0.31 | 39 |
| micro avg | 0.58 | 0.60 | 0.59 | 2597 |
| macro avg | 0.63 | 0.53 | 0.56 | 2597 |
| weighted avg | 0.61 | 0.60 | 0.58 | 2597 |

# משימת ביאור

לכל לייבל, היה משימת ביאור משלה.
עבור ה **skills**:
השתמשנו בכלי שנקרא SkillsNer:
מערכת זיהוי ישות (NER) שאומנה עם מכונת למידה וקטורית (SVM) על קורפוס של יותר מ 5000 מאמרים מדעיים.
הם פיתחו את מערכת זו על ידי מדידת הביצועים של הגישה שלהם מול אימון מודלים שונים והאימות תוצאות יחד עם צוות פסיכולוגים.
התוצאות של SkillsNer הם F1-score של 72.6.

עבור ה **experience**:
השתמשנו ב regex matcher, חילצנו את כל התווים בהם מופיע המילה 'year'.

עבור ה **education**:
עברנו על 2 datasets אשר מכילים שדות של education,הוצאנו אותם, וחיברנו את כולם תחת אותה קבוצה, וגם פה, השתמשנו ב regex matcher כדי לחלץ את ה education.

יש צורך לציין כי ה"extractors" יכלו להיות יותר טובים וכי יש מצב שפספסנו פה מילים.

יכולנו להשתמש בכלים יותר חזקים לזיהוי skills,experience,education שיכולים להוציא תוצאות יותר טובות.

# קורפוס

קורפוס תיאורי התפקיד שלנו הגיע מKaggle5, שהוא זמין לציבור.
נראה שהם הוצאו מפורטלי עבודה מקוונים מתפקידים בבריטניה.
מגוון רחב של תעשיות מיוצגים, כגון מחלקות HighTech,פיננסים, שירותי בריאות ומכירות.

# גודל ותפוצה

לאחר מעבר על ה-דאטה סט הזה, והמעבר לIOB format.
הגענו לגודל קובץ של:10MB
train + 3086 dev + 3096 test sentences 14999
ב train יש 14999 שורות.
ב dev יש 3086 שורות.
ב test יש 3096 שורות.

# עיבוד נתונים מראש

כל הישויות סומנו באמצעות IOB.
טעות היא דבר בלתי נמנע במשימות תיוג אנושיות.
אפשרי למתן זאת בכמה היבטים:

1. סיווג מחדש של education על ידי בניית classification, עבור התווית של ה education בלבד.
2. סיווג מחדש של ה experience על ידי בניית classification, עבור התווית של ה experience בלבד.
3. שימוש באלגוריתם עם תוצאות יותר טובות מ SkillsNer לטובת המשימה.

# מודלים

## LM-LSTM-CRF

### הסבר

שיטות חילוץ מסורתיות השתמשו ב-מודלים של למידת מכונה כמו HMM ו CRF והשיגו ביצועים גבוהים יחסית.
עם זאת, על שיטות אלו יש הסתמכות רבה על תכונות בעבודת יד (למשל, אם מילה היא באותיות רישיות) ומשאבים ספציפיים לשפה (למשל עיתונים וכו').
לכן, יכול מאוד להיות שיהיה קשה ליישם אותם במשימות חדשות או להריץ אותם בדומיינים חדשים.
כדי להתגבר על החיסרון הזה,הוצעו רשתות עצביות (NNs) למשימת חילוץ אוטומטי של תכונות במהלך למידת מודל.
עם זאת, בהתחשב במספר המכריע של פרמטרים ב NNs וגודל קטן של תיוג רצף קורפוס לא יכול להיות מספיק כדי להכשיר מודלים מסובכים.
כך, הדרך של תהליך הלמידה עם ידע נוסף יכול להיות בחירה נבונה.

לדוגמא: ניתן לשפר את NER על ידי ביצוע משימות אחרות כמו קישור ישויות.

רק לאחרונה, ידע ברמת ה-character-level אומת כדי לעזור במשימה של תיוג רצפים.
עם זאת, הידע שחולץ דרך אימון מקדים אינו ספציפי למשימה, ולכן מכיל חלק גדול לא רלוונטי.
אז גישה זו תדרוש מודל וקורפוס חיצוני גדול יותר.

לכן LM-LSTM-CRF זה מודל הממנף תיוגים ברמת המילים וגם ברמת ה character-level בצורה יעילה.
במודל זה מעסיקים רשתות כבישים מהירים (Srivastava,Greff and Schmiduhuber 2015).
כדי להפוך את הפלט של השכבות למרחבים סמנטיים שונים, ובכך לתווך ולאחד אותם לשתי משימות.
לידע ברמת המילה - בוחרים לכוון הטמעת מילים מאומנות מראש במקום אימון משותף או אימון מקדים של כל השכבות ברמת המילה.
מכיוון שרוב הפרמטרים בשכבות ברמת המילה מגיעים משכבת ההטמעה ואימון משותף או אימון מקדים, זה עולה הרבה זמן ומשאבים.
אסטרטגיית האימון המשותף שלהם מאפשרת להם ללכוד ידע שימושי יותר עם רשת קטנה יותר, ובכך להניב הרבה יעילות טובה יותר ללא אובדן יעילות.
המאמר: https://arxiv.org/pdf/1709.04109.pdf

## הטמעת הקוד

לקחנו את הקוד של מודל זה (https://github.com/LiyuanLucasLiu/LM-LSTM-CRF/tree/master)
הטמענו אותו אצלנו בקוד, עשינו התאמות לסביבת עבודה שלנו (macbook pro m1).

## הריצה

הרצנו את המודל עם ה datasets שיצרנו (job-descriptions).
כחלק מהריצה, בשביל לאמן את המודל, הרצנו אותו על 35 סיבובים (epoc).
כל פעם ה loss ירד והf1 score עלה.
הריצה לקחה 15 שעות.

```
(loss: 2.8502, epoch: 1, dev F1 = 0.8544, dev acc = 0.9704, F1 on test = 0.8326, acc on test= 0.9633), saving...
epoch: 1     in 200 take: 5383.1402270793915 s
DEV : total : dev_f1: 0.8586 dev_rec: 0.8046 dev_pre: 0.9203 dev_acc: 0.9723 |

DEV : O : dev_f1: 0.9867 dev_rec: 0.9946 dev_pre: 0.9788 dev_acc: 0.0000 | {'I-SKILL': 87, 'B-SKILL': 276}

DEV : B-SKILL : dev_f1: 0.9155 dev_rec: 0.8942 dev_pre: 0.9379 dev_acc: 0.0000 | {'O': 596, 'I-SKILL': 151}

DEV : I-SKILL : dev_f1: 0.4982 dev_rec: 0.3809 dev_pre: 0.7197 dev_acc: 0.0000 | {'B-SKILL': 142, 'O': 851}

DEV : B-EDU : dev_f1: 0.2000 dev_rec: 0.1111 dev_pre: 1.0000 dev_acc: 0.0000 | {'O': 8}

TEST : total : test_f1: 0.8500 test_rec: 0.7972 test_pre: 0.9102 test_acc: 0.9676 |

TEST : O : test_f1: 0.9845 test_rec: 0.9932 test_pre: 0.9759 test_acc: 0.0000 | {'B-SKILL': 382, 'I-SKILL': 72}

TEST : B-SKILL : test_f1: 0.9029 test_rec: 0.8911 test_pre: 0.9150 test_acc: 0.0000 | {'O': 667, 'I-SKILL': 140}

TEST : I-SKILL : test_f1: 0.4544 test_rec: 0.3290 test_pre: 0.7340 test_acc: 0.0000 | {'O': 965, 'B-SKILL': 228}

TEST : B-EDU : test_f1: 0.1000 test_rec: 0.0526 test_pre: 1.0000 test_acc: 0.0000 | {'O': 15, 'B-SKILL': 3}

(loss: 2.3695, epoch: 2, dev F1 = 0.8586, dev acc = 0.9723, F1 on test = 0.8500, acc on test= 0.9676), saving...
epoch: 2     in 200 take: 7407.2884957790375 s
DEV : total : dev_f1: 0.8705 dev_rec: 0.8187 dev_pre: 0.9294 dev_acc: 0.9738 |

DEV : O : dev_f1: 0.9872 dev_rec: 0.9953 dev_pre: 0.9792 dev_acc: 0.0000 | {'I-SKILL': 147, 'B-SKILL': 173}

DEV : B-SKILL : dev_f1: 0.9241 dev_rec: 0.8973 dev_pre: 0.9526 dev_acc: 0.0000 | {'O': 618, 'I-SKILL': 107}

DEV : I-SKILL : dev_f1: 0.5248 dev_rec: 0.4121 dev_pre: 0.7224 dev_acc: 0.0000 | {'B-SKILL': 142, 'O': 801}

DEV : B-EDU : dev_f1: 0.2000 dev_rec: 0.1111 dev_pre: 1.0000 dev_acc: 0.0000 | {'O': 8}

TEST : total : test_f1: 0.8533 test_rec: 0.7958 test_pre: 0.9199 test_acc: 0.9686 |
```

### התוצאות

הגענו לתוצאות יחסית מרשימות של 0.9246 אחרי 15 שעות ריצה.

| Dataset | Model | Epoc | F1 | Time(h) |
|---|---|---|---|---|
| **Job-description** | LM-LSTM-CRF | 35 | 0.9246 | 15 |

## Flair

### הסבר

משפחה גדולה של משימות NLP כגון זיהוי ישויות בשם NER ותיוג חלקי דיבור POS עשויות להתנסח כבעיות תיוג רצף.
מתייחסים לטקסט כרצף של מילים שיש לתייג עם תגים לשוניים.
גישות עדכניות לתיוג רצף בדרך כלל משתמשות ב LSTM שזה גרסה של רשתות עצביות חוזרות דו-כיווניות ושדה אקראי מותנה לאחר מכן CRF.
מרכיב מכריע בגישות כאלה הוא הטמעת מילים, בדרך כלל מאומנת על אוסף גדול מאוד של נתונים לא מסומנים כדי לסייע בלמידה והכללה.
ב flair השתמשו בכמה גישות חדשות:

1. Contextual string embeddings
   - אימון על קורפוסים לא מתויגים.
   - ללכוד את משמעות המילה דרך ההקשר, ובכך מייצרים embeddings שונים בהקשרים שונים.
   - התמודדות עם מילים נדירות ו-איות שגוי.
2. Neural character-level language modeling

- ביסוס ה embeddings הם על המודלים הכי חדישים של ה NNs, שאפשר לעצב שפה לפי קוונסות של תווים ולא של מילים.
- ייחוס ל LSTM-LM פונקציות סמנטיות ספציפיות, כגון חיזוי סנטימנט ללא אימון וזה גורם לניצול מבחר מתאים של מצבים נסתרים ממודל שפה ליצור embeddings ברמת המילה שהן יעילות יותר למשימות תיוג.

3. State-of-the-art sequence labeling

כל משפט מועבר כרצף של תווים לרמת דו-כיוונית של מודל שפה עצבית, שממנו הם שואבים עבור כל מילה את מצבי התו הפנימי כדי ליצור embeddings (מחרוזות קשורות זו בזו).

לאחר מכן, המודל Bi-LSTM-CRF משתמש ב embeddings האלו

## הטמעת הקוד

השתמשו בספרייה flair שקיימת ב python, ועשינו fine tune.

## הריצה

הרצנו את המודל עם ה datasets שיצרנו (job-descriptions).
כחלק מהריצה, בשביל לאמן את המודל, הרצנו אותו על 15 סיבובים (epoc).
כל פעם ה loss ירד והf1 score עלה.
הריצה לקחה 15 שעות.

```
2023-06-03 10:35:30,898 epoch 1 - iter 0/566 - loss 166.34434509 - samples/sec: 115.72
2023-06-03 10:44:16,001 epoch 1 - iter 56/566 - loss 21.31661368 - samples/sec: 3.41
2023-06-03 10:52:22,960 epoch 1 - iter 112/566 - loss 16.29596274 - samples/sec: 3.68
2023-06-03 11:02:04,450 epoch 1 - iter 168/566 - loss 14.16415755 - samples/sec: 3.08
2023-06-03 11:10:54,881 epoch 1 - iter 224/566 - loss 12.98463648 - samples/sec: 3.38
2023-06-03 11:19:05,705 epoch 1 - iter 280/566 - loss 11.99075442 - samples/sec: 3.65
2023-06-03 11:27:33,767 epoch 1 - iter 336/566 - loss 11.22239269 - samples/sec: 3.53
2023-06-03 11:35:41,562 epoch 1 - iter 392/566 - loss 10.63254863 - samples/sec: 3.67
2023-06-03 11:44:23,673 epoch 1 - iter 448/566 - loss 10.14448355 - samples/sec: 3.43
2023-06-03 11:51:52,684 epoch 1 - iter 504/566 - loss 9.73219645 - samples/sec: 3.99
2023-06-03 12:01:26,362 epoch 1 - iter 560/566 - loss 9.44994843 - samples/sec: 3.12
2023-06-03 12:01:53,881 ----------------------------------------------------------------------------------------------------
2023-06-03 12:01:53,882 EPOCH 1 done: loss 9.4151 - lr 0.1000
2023-06-03 12:01:53,882 BAD EPOCHS (no improvement): 0
2023-06-03 12:01:53,883 ----------------------------------------------------------------------------------------------------
2023-06-03 12:02:00,453 epoch 2 - iter 0/566 - loss 9.24654770 - samples/sec: 272.91
2023-06-03 12:06:08,964 epoch 2 - iter 56/566 - loss 6.13980147 - samples/sec: 7.21
2023-06-03 12:09:56,521 epoch 2 - iter 112/566 - loss 5.87939421 - samples/sec: 7.88
2023-06-03 12:14:06,385 epoch 2 - iter 168/566 - loss 5.77996976 - samples/sec: 7.17
2023-06-03 12:19:21,265 epoch 2 - iter 224/566 - loss 5.90563866 - samples/sec: 5.69
2023-06-03 12:23:03,750 epoch 2 - iter 280/566 - loss 5.75470873 - samples/sec: 8.06
2023-06-03 12:27:49,443 epoch 2 - iter 336/566 - loss 5.72263122 - samples/sec: 6.27
2023-06-03 12:31:45,242 epoch 2 - iter 392/566 - loss 5.63268232 - samples/sec: 7.60
2023-06-03 12:36:07,011 epoch 2 - iter 448/566 - loss 5.54249114 - samples/sec: 6.85
2023-06-03 12:39:51,732 epoch 2 - iter 504/566 - loss 5.45525699 - samples/sec: 7.98
2023-06-03 12:43:30,972 epoch 2 - iter 560/566 - loss 5.41222047 - samples/sec: 8.17
2023-06-03 12:43:50,124 ----------------------------------------------------------------------------------------------------
2023-06-03 12:43:50,124 EPOCH 2 done: loss 5.4046 - lr 0.1000
2023-06-03 12:43:50,125 BAD EPOCHS (no improvement): 0
```

## התוצאות

| Dataset | Model | Epoc | F1 | Time(h) |
|---|---|---|---|---|
| **Job-description** | Flair | 15 | 0.8843 | 19.995833 |

# תוצאות

| Dataset | Model | Epoc | F1 | Time(h) |
|---|---|---|---|---|
| **Job-description** | LM-LSTM-CRF | 35 | 0.9246 | 15 |
| **Job-description** | Flair | 15 | 0.8843 | 19.995833 |

כמו שאפשר לראות באיור למעלה, המודל LM-LSTM-CRF שרץ 15 שעות עם 35 סיבובים, עשה תוצאות יותר טובות מאשר Flair שרץ על 15 סיבובים, אבל רץ יותר זמן (כמעט 20 שעות).

# סיכום

במאמר זה, הצגנו דרך יותר יעילה, יותר טובה לפתור את בעיית הזיהוי וחילוץ ישויות מרכזיות טקסטים מבוססי job descriptions.
לקחנו דאטה סטים של job descriptions, הפעלנו כמה אלגוריתמים כדי להוציא entities:
skills, education, experience.
ויצרנו דאטה סט חדש בפורמט IOB, עם התגיות האלו.
הרצנו על Flair ועל ה LM-LSTM-CRF והשגנו תוצאות.
על פי התוצאות נראה כי ה LM-LSTM-CRF מביא תוצאות יותר טובות מאשר ה Flair.

# הפניות


https://aclanthology.org/2022.lrec-1.128.pdf
https://github.com/LiyuanLucasLiu/LM-LSTM-CRF/tree/master
https://arxiv.org/pdf/1709.04109.pdf
https://www.kaggle.com/datasets/airiddha/trainrev1
https://drive.google.com/file/d/17yVpFA7MmXaQFTe-HDpZuqw9fJlmzg56/view
https://github.com/flairNLP/flair