**הבנת שפה טבעית**

**אביב תשפ"ג**

תוכן עניינים

[מודלי N-grams 4](#_Toc140608957)

[מודל שפה מבוסס רשת נוירונים 7](#_Toc140608958)

[וקטורים "דלילים" 7](#_Toc140608959)

[וקטורים "צפופים" 7](#_Toc140608960)

[שימוש במודל feed-forward 8](#_Toc140608961)

[RNNS 10](#_Toc140608962)

[מודל שפה מבוסס RNN 10](#_Toc140608963)

[RNN למטלות NLP נוספות 10](#_Toc140608964)

[שיפורים ל-RNN 10](#_Toc140608965)

[Encoder-Decoder RNN 11](#_Toc140608966)

[Attention 12](#_Toc140608967)

[Transformers 12](#_Toc140608968)

[Semantic Role Labeling (SRL) 15](#_Toc140608969)

[קורפוסים ידועים 15](#_Toc140608970)

[Proposition Bank (Palmer et al., 2005) 15](#_Toc140608971)

[NomBank (Meyers et al., 2004) 16](#_Toc140608972)

[FrameNet (Baker et al., 1998) 16](#_Toc140608973)

[Semantic Role Labeling (SRL) 17](#_Toc140608974)

[תחביר 17](#_Toc140608975)

[SRL באמצעות רשתות 18](#_Toc140608976)

[SemBERT 18](#_Toc140608977)

[מטלות שונות 19](#_Toc140608978)

[חילוץ מאורעות 20](#_Toc140608979)

[Named Entity Recognition 21](#_Toc140608980)

[Relation Extraction 22](#_Toc140608981)

[Coreference 23](#_Toc140608982)

[Complex Event Identification 23](#_Toc140608983)

[Question Answering 24](#_Toc140608984)

[Extractive QA 24](#_Toc140608985)

[Extractive QA with IDK 24](#_Toc140608986)

[Yes/No QA 25](#_Toc140608987)

[QAMR 25](#_Toc140608988)

[Textual Entailment (גרירה טקסטואלית) 26](#_Toc140608989)

[Prompting for NLU 26](#_Toc140608990)

[Proxy-Task Based Learning 26](#_Toc140608991)

[Template Based Learning 29](#_Toc140608992)

[Instruction Based Learning 30](#_Toc140608993)

[QA as a Meaning Representation 32](#_Toc140608994)

[UCCA Annotation 33](#_Toc140608995)

[מודל ה-UCCA 33](#_Toc140608996)

[אלגוריתם תיוג UCCA 36](#_Toc140608997)

מודלי N-grams

נקודת המפנה הראשונה בתחום ה-NLP הייתה שימוש במודלים סטטיסטיים לטובת עיבוד השפה. כאשר משתמשים בהסתברות ניתן לענות על שאלות כגון: עד כמה סביר ניסוח מסוים של משפט, בהינתן התחלה של משפט מהי ההסתברות למילה הבאה וכדו'. המטרה של **מודל שפה** היא בניית מודל הסתברותי של שפה טבעית. שימושים מידיים שניתן לעשות למודל כזה הם למשל השלמת מילים בזמן הקלדה, תיקון שגיאות כתיב או שגיאות תחביר וכדו'. עם זאת, מודלים כאלה כיום מהווים בסיס לכמעט כל הבעיות בעיבוד שפה טבעית. כעת נתייחס לשאלה כיצד ניתן ללמוד מודל כזה של הסתברויות השפה.

ראשית נגדיר מהו token. Token הוא מילה או חלק מילה. הגדרה של "מילון ה-token-ים" נעשה פעם אחת, ולאחר מכן ניתן לבצע "טוקניזציה" – כלומר חלוקה של רצפים (משפטים) ל-tokens. ישנם כמה אפשרויות לבניית מילון ה-tokens, למשל ניתן להגדיר כל תו כ-token, כל מילה, חלקי מילים. החלוקה נקבעת לפי המשימה שאותה רוצים לפתור.

מודל השפה הפשוט ביותר מבוסס N-gram. N-gram הוא רצץ של N tokens. ספציפית, unigram הוא רצף של token אחד, bigram – שתי tokens ו-trigram שלוש tokens. מודל unigram ממדל הסתברות של כל מילה בנפרד, מודל bigram בודק את ההסתברות של כל זוגות המילים במשפט וכן הלאה. ככל ש-N גדול יותר המודל יכול למדל בצורה טובה יותר את השפה כרצף tokens. מצד שני, ככל ש-N גדול יותר, כל רצף של N tokens הוא נדיר יותר ולכן קשה לדעת מהי ההסתברות של אותו רצף. לדוגמה, בהינתן התחלה של משפט: "can you give me a cup of" – אם נרצה שמודל השפה יחזה מהי המילה הבאה, במודל unigram המודל פשוט יתן את המילה הנפוצה ביותר בקורפוס, למרות שהיא לא קשורה למילים הקודמות, מודל bigram יבדוק את כל זוגות המילים שהמילה הראשונה היא of: (of, \_), מודל trigram יבדוק את כל שלישיות המילים המתחילות ב-cup of: (cup, of,\_) ויבחר את השלישיה המופיעה הכי הרבה.

פורמלית יותר, במודל bigram אנו מעוניינים לדעת מהי המילה הבאה בהינתן המילה הקודמת:   
כאשר היא מספר הפעמים שבהם הזוג מופיע בקורפוס, ו- הוא מספר הפעמים ש- מופיע. במילים אחרות הוא אחוז הפעמים שבהם הוא ה-token שמופיע אחרי .

דבר נוסף שניתן לעשות עם מודל שפה הוא לחשב את **ההסתברות של משפט מסויים**, כלומר כמה סביר שיהיה בקורפוס משפט כזה. את ההסתברות ניתן לחשב באמצעות הנוסחה:  
ההנחה מאחורי שימוש ב-N-gram היא שהמידע על ההסטוריה שלפני N tokens לא רלוונטית, למשל עבור bigram:

בדומה לכך, ניתן להגדיר עבור מודל trigram:

*וכן עבור N-gram כללי:*

במקרה כזה הסתברות למילה מסוימת תלויה ב- המילים שלפניה.

שיטה זו של N-grams מבוססת על maximum likelihood. הסיבה לך היא ש-ML מחפשת את ההתפלגות שתמקסם את ההסתברות לקבל את הדגימות שדגמנו. בהינתן דגימות, ההתפלגות שעבורה ההסתברות לקבל את אותן דגימות מקסימלית, היא ההתפלגות הנ"ל (כלומר מספר הדגימות שהתקבלו בפועל חלקי כל הדגימות). נראה הוכחה ספציפית ל-bigram model כי המודל שהוצג הוא אכן פתרון ה-maximum likelihood. נניח משפט באורך N ומילון באורך n מילים . לצורך ההוכחה לא נשתמש ב-token מיוחד לתחילת המשפט, אלא נחשב בנפרד את ההסתברות של המילה הראשונה ושל שאר המשפט. ניתן לבטא את ההסתברות בצורה הבאה.

כאשר הוא אינדיקטור אם היא המילה הראשונה במשפט, ו- הוא מונה (counter) לכמות הפעמים ש- מופיע במשפט.

*המטרה היא למצוא את המקסימום של ההתפלגות תחת האילוץ:*

*ניתן להשתמש בכופלי לגראנז':*

*נגזור את הלגראנז'יאן לפי כל אחד מהמשתנים:*

*בנוסף, קיים האילוץ:*

*כפי שהסקנו לעיל:*

*כלומר, מודל ה-bigram הינו maximum likelihood. (מש"ל וכו'). ניתן לגזור זאת גם למקרה הכללי של N-gram.*

*כעת נדון בשאלה כיצד ניתן לבצע אבלואציה של מודל שפה. קיימות שתי גישות, אבלואציה אינטרינזית ואקסטרינזית. האבלואציה האקסטרינזית מתבססת על אבלואציה של מטלות (tasks) המבוססות על מודל השפה. למשל, פיתוח מודל question answering או של תרגום (machine translation) המבוסס על מודל השפה. בצורה כזאת, ניתן לכמת את הצלחת מודל השפה במטלות אלה, ובעזרתן להסיק את טיב מודל השפה. הייתרון בגישה זו הוא שאם המטרה הסופית של המפתח היא להשתמש במודל השפה למטלה זו, אבלואציה כזאת משקפת בצורה טובה לאילו ביצועים ניתן לצפות. החיסרון העיקרי בגישה זו הוא שהבדיקה אינה ישירה אלא עקיפה.*

*הגישה האינטרינזית מנסה לתת מדד כמותי לטיב המודל בצורה ישירה באמצעות ההסתברויות של המודל. השיטה הנפוצה ביותר היא ה-perplexity (PP). בשיטה זו, המפתח שומר test set עליו הוא לא מאמן את מודל השפה. לאחר האימון מתבצעת בדיקה של ההסתברויות שמודל השפה נותן למשפטים מתוך ה-test set. ההנחה היא שעבור מודל שפה טוב ההסתברויות תהינה גבוהות. ספציפית, מדד ה-PP הינו ביחס הפוך להסתברות שהמודל נותן:*

*למשל, עבור הנחת bigram מתקבל:*

*ככל שמדד ה-perplexity נמוך, כך ההסתברות למשפט גבוהה, מה שמעיד על מודל שפה איכותי יותר. הייתרון בשיטה זו הוא שניתן לבדוק את טיב המודל בצורה "אינטרינזית" בלי להסתמך על מטלות עקיפות אחרות. חיסרון בשיטה זו הוא שלא תמיד הוא משקף כמה המודל יהיה טוב במטלות אחרות. חיסרון נוסף בשיטה זו הוא שבמקרה ש-N-gram מסויים לא מופיע ב-train set ההסתברות שלו תהיה שווה לאפס, וה-perplexity יהיה אינסופי. פיתרון אפשרי לבעיה זו הוא שבמקרה שבו N-gram לא קיים נתבסס על ה-(N-1)-gram. למשל, במקרה ששלישיית מילים לא נמצאת ב-train set אך נמצאת ב-test set ניתן לחלק אותה לשני זוגות מילים. אם גם הזוגות לא נמצאים ניתן לחלק ל-unigrams.*

*לסיכום, מודל שפה ממדל את ההסתברויות של שפה טבעית, הן הסתברות של משפט שלם והן הסתברות של המילה הבאה במשפט. מודל N-gram מניח תלות סטטיסטית של מילה במשפט רק ב-N מילים שלפניה. תחת הנחה זו, ניתן למצוא משערך maximum likelihood על בסיס ספירה של N-grams הקיימים בקורפוס. שיטה זו היא בסיסית ופשוטה אך יכולה לתת ביצועים טובים במשימות פשוטות.*

מודל שפה מבוסס רשת נוירונים

*הערה: בשיעור המרצה העמיק במבוא לרשתות נוירונים (פרספטרון, פונקציות אקטיבציה, ועוד). לא נסקור מבוא זה בסיכום.*

*הרשת הפשוטה ביותר בה ניתן להשתמש כמודל שפה היא feedforward network. רשת כזאת תקבל בכניסה ייצוג של N מילים (בדומה ל-N-gram) ותחזה את המילה הבאה. הייתרון של מודל זה על פני N-gram הוא שהעיבוד נעשה על וקטורי embedding של מילים, ולא המילים עצמן. דבר זה מאפשר הכללה יותר טובה, למשל המודל יוכל לחזות בצורה טובה גם רצפים שלא הופיעו בקורפוס האימון, אם היו קיימים רצפים דומים ב-embedding שלהם. למשל, במשפט הבא:*

*I forgot to make sure that the dog gets…*

*המילה "fed" צריכה להיות בעלת הסתברות גבוהה. במודל שפה מבוסס N-grams, אם הרצף לא היה קיים (למשל ב-trigram – (dog gets fed)) המודל לא ידע להשלים את המשפט. לעומת זאת, במודל מבוסס רשת אשר מתבסס על embeddings, במידה והיה קיים בקורפוס המשפט הבא:*

*I forgot to make sure that the cat gets fed.*

*המודל ידע להשלים את המשפט בהסתברות גבוהה, מכיוון שה-embeddings של dog ושל cat קרובים. הרעיון ב-embeddings הוא שניתן לייצג מילים במרחב וקטורי ולהגדיר משמעות של מילה באמצעות ההתפלגות הסטטיסטית של השפה במרחב זה. למשל מילים בעלות משמעות סמנטית או תחבירית דומה יהיו קרובות במרחב זה. רעיון זה מופיע עוד בחיבור של Osgood et al., 1957 אשר מייצג משמעות של מילה כנקודה במרחב. בפרסום המקורי המחבר התייחס לשלושה ממדים של משמעות:*

* ***Valence*** *– עד כמה המילה "נעימה"*
* ***Arousal*** *– כמה "רגש" יש במילה*
* ***Dominance*** *– כמה "שליטה" יש במילה (ציווי)*

עם זאת, כיום משתמשים בוקטורים מרובי ממדים לצורך אפיון של מילה.

קיימים שני סוגי embeddings – "דלילים" (sparse) ו"צפופים" (dense).

וקטורים "דלילים"

הוקטורים ה"דלילים" מיוצגים על ידי ספירה של מילים אשר מופיעות בקרבת המילה בקורפוס. בצורה טיפוסית, הוקטור יהיה בממד שגודלו כגודל המילון, כאשר כל אבר מייצג מילה אחרת, והערך המופיע בו הוא counter של כמות הפעמים שמילה זו מופיעה בקרבת המילה שאותה מקודדים. צורה זו היא "דלילה" מכיוון שלרוב, רוב הערכים בוקטור יהיו אפסים (או לפחות נמוכים מאוד), למשל, הסיכוי שתופיע המילה "אנציקלופדיה" ליד המילה "בלגיה" נמוך מאוד.

בשיטת word counts, מחושבת טבלה בגודל כאשר הוא מספר המילים במילון. איבר בטבלה מכיל את מספר הפעמים שמילה מופיעה במרחק מסוים קבוע (למשל 6 מילים) ממילה . מכיוון שיחס זה הוא גם הפוך, כלומר אם מילה קרובה ל- במקום מסוים בקורפוס אז גם מילה קרובה ל- באותו המקום, המטריצה סימטרית (). המטריצה נקראת word-word matrix או term-context matrix. שתי מילים תחשבנה קרובות במרחב ה-embeddings אם שורה דומה (למשל קרובה בנורמת ) לשורה .

וקטורים "צפופים"

שיטות לייצוג "צפוף" משתמשות בוקטור אשר הממד שלו קטן משמעותית מגודל המילון. בשיטות אלו נדרש להשתמש בשיטות חכמות יותר על מנת לקודד את המידע על המילה בצורה "דחוסה" יותר. הדוגמה המפורסמת ביותר לשיטה כזו היא Word2Vec. הרעיון בשיטה זו הוא לאמן מסווג שיחזה האם מילה תופיע בקרבת מילה . בצורה זו, ניתן להשתמש בקורפוס על מנת ליצור dataset גדול של זוגות מילים בעלות תיוג "חיובי" ללא צורך בתיוג (self-supervised), וכל המילים שאינן מופיעות ביחד תהיינה דוגמאות שליליות. לאחר מכן, מאומן מודל logistic regression שיחזה את המילים החיוביות, כאשר הפרמטרים הנלמדים הם הייצוגים של המילים. לאחר האימון, ייצוגים אלה נלקחים כ-embeddings.

*ישנם שני אלגוריתמים שהוצגו במאמר של Word2Vec: skip-gram ו-Continuous Bag of Words (CBOW).ב-skip-gram מנסים לחזות את סביבות המילה על בסיס המילה האמצעית (למשל עבור סביבה של – את על בסיס ), וב-CBOW להיפך – מנסים לחזות את המילה האמצעית () בהינתן סביבות המילה (). בשתי השיטות, לאחר למידת מודל כזה מתקבל ייצוג וקטורי של כל מילה במילון.*

*אחת מהאפשרויות לחישוב ההסתברויות היא מהצורה הבאה:  
כלומר, חישוב softmax בין ייצוג הווקטור הנתון לבין כל הוקטורים במילון . במהלך האימון, תתבצע אופטימיזציה כך שההסתברות תהיה גבוהה עבור זוגות מילים אשר קיימים בקורפוס.*

*הבעיה בשיטה זו היא שעבור כל חישוב של אנו נדרשים לחשב את המכפלות של עם הווקטורים המייצגים של כל המילים במילון. חישוב זה הינו אינטנסיבי עבור מילון גדול.* ***Negative sampling*** *היא שיטה הנועדה לפתור את הבעיה (נקראת גם SGNS - Skip-gram with Negative Sampling). Mikolov et al., 2013 הציעו את הרעיון הבא, במקום לחשב את ההסתברות על פני כל המילים במילון, ניתן לדגום k מילים אקראיות ולהשתמש רק בהן כדוגמאות שליליות. בצורה כזאת, הדמיון בין הווקטורים החיוביים יתקרב, והמרחק בין הווקטורים השליליים יתרחק בממוצע, מכיוון שהדוגמאות השליליות נדגמות באקראי. פונקציית המטרה אשר הוצגה במאמר היא:  
כאשר היא פונקציית הסתברות שנקבעה על בסיס ניסויים. היתרונות באלגוריתם הם מהירות האימון שלו והיעילות שלו. בנוסף, קיימים מימושים רבים שלו באינטרנט וכן "מילוני embeddings" מאומנים שלו.*

*נעיר כי בשיטות ה-Word2Vec כל מילה (או token) מקבלת ייצוג סטטי, אשר אינו תלוי בקונטקסט. עובדה זו יכולה להוות בעיה למשל כאשר יש למילה מספר משמעויות. לפעמים חשוב לתת למילה ייצוג לפי המשמעות שלה כתלות בקונטקסט. למשל היינו רוצים כי הייצוג של המילה "orange" במשפטים "A tasty orange" ו-“An orange pillow” יהיה שונה מכיוון שהמשמעות שונה. קיימים מודלים אחרים אשר באים להתמודד עם בעיה זו. למשל BERT נותן ייצוג לכל מילה על פי הקונטקסט של המשפט.*

*אחד מהתכונות המעניינות בווקטורים הנלמדים מ-Word2Vec היא הלינאריות של ה-embeddings (parallelogram model). למשל, הראו במאמר מוקדם בתחום כי:  
דבר זה מעיד כי ל-embeddings הנלמדים ייצוגים לינאריים בקירוב.*

*לסיכום, היתרונות של מודל שפה נלמד מבוסס feed forward אל מול N-gram:*

* *קל יותר למדל רצפים ארוכים של tokens*
* *הכללה טובה יותר של מילים דומות קונטקסטואלית (הדוגמה שהובאה לעיל עם cat ו-dog)*
* *ביצועים טובים יותר בחיזוי מילה*

*חסרונות:*

* *מימוש מורכב יותר, נדרש ללמוד מודל, לעומת N-gram בו רק נדרש לספור מילים בקורפוס*
* *איטי יותר ב-inference (לעומת N-gram בו מספיק להשתמש ב-lookup table)*
* *יותר אנרגיה לאימון*
* *פחות אינטרפרטבילי – קשה להבין את משמעות הווקטורים הנלמדים*

שימוש במודל feed-forward

*הארכיטקטורה של מודל שפה מסוג feed-forward הוא יחסית פשוט. ניקח לדוגמה מודל שמטרתו לחזות את ה-token בהינתן שלושת ה-tokens האחרונים: . נניח שנתון וקטור embeddings עבור כל token במילון , כאשר הוא הממד של וקטור ה-embeddings. בנוסף, נניח כי מספר ה-tokens במילון הוא . מכל הוקטורים . ניצור מטריצה כך שכל עמודה במטריצה מייצגת את ה-token ה- במילון ומכילה את ה-embeddings של אותה המילה. כעת, נייצג את ה-tokens של שלושת המילים האחרונות במשפט שאותו מעוניינים לחזות באמצעות one hot encoding, כלומר וקטור בגודל המילון המכיל אפסים בכולו מלבד המילה הנוכחית. את המילים המקודדות כ-one hot נייצג באמצעות . נשים לב כי מתקיים:  
כלומר, אם נכפיל את מטריצה ה-embeddings בייצוג ה-one hot של ה-token, נקבל את ה-embeddings של ה-token הנבחר. לאחר קבלת מתבצע שרשור (concatenation) של שלושת הווקטורים לווקטור embeddings אחד: :  
את הווקטור הזה מעבירים בשכבת fully connected:  
כאשר מספר העמודות ב- הוא , ומספר השורות הוא כמספר הנוירונים בשכבה. לבסוף מעבירים את לשכבת המוצא:  
הווקטור יהיה וקטור שגודלו כגודל המילון ומכיל את ההסתברויות של כל אחת מהמילים להיות המילה הבאה. סכמה של התהליך מוצגת באיור 1.*

**

איור 1: סכמה של מודל שפה מסוג feed-forward. כל מילה ממופית לאינדקס שלה במילון, לאחר מכן מומרת ל-one hot vector. לאחר מכן מוכפלת במטריצה E כך שמתקבלים וקטורי ה-embeddings של המילים. וקטורים אלו משורשרים ועוברים בשכבת fully connected ושכבת output, כך שמתקבל בסופו של דבר וקטור הסתברויות המכיל את ההסתברות לכל אחת מהמילים במילון.

*ניתן לאמן מודל כזה בצורה פשוטה כ-self supervised. בהינתן קורפוס, ניתן להתייחס לכל רביעיית מילים כדגימות (שלושת המילים הראשונות) ותיוג (המילה האחרונה). האימון מתבצע באמצעות cross entropy loss באמצעות gradient descent. בהרצאות כל משוואות ה-loss function וה-backpropagation מוצגים בהרחבה, אך לא נרחיב בסיכום זה.*

*לבסוף, קיימות שתי גישות לאימון מודל כזה. גישה אחת משתמשת בווקטורי embeddings קבועים (pre-trained) למשל, באמצעות Word2Vec, והאימון מתבצע אך ורק עם הפרמטרים של הרשת (). לעומת זאת, בגישת ה-end-to-end מתבצע backpropagation גם לווקטורי ה-embeddings, כך שהאימון מתבצע גם על הייצוגים של ה-tokens. נעיר כי קיימות גישות היברידיות אשר מאמנות בתחילה על וקטורי embeddings קבועים, ורק בסוף מבצעות fine tuning על כל הרשת כולל הייצוגים.*

RNNS

מודל שפה מבוסס RNN

*הערה: בשיעור המרצה העמיק בארכיטקטורת RNN. לא נסקור מבוא זה בסיכום.*

*רשת RNN מעבדת רצפים של inputs, גם אם האורך שלהם לא קבוע, על ידי וקטור "זיכרון" . תכונה זו הופכת את ה-RNN למתאימה באופן טבעי לעיבוד שפה טבעית, מכיוון שבאופן עקרוני, יכול להכיל את כל ההיסטוריה של הרצף . בשונה מרשת feed-forward לא נדרש להגדיר בצורה מפורשת את אורך ההיסטוריה הרצויה, כלומר כמה מילים להכניס לרשת. משוואות הרשת הן מהצורה הבאה:*

*עם מודל כזה אפשר להעריך בצורה ישירה את ההסתברות של המילה הבאה:  
וכן את ההסתברות לרצף של מילים:*

*אימון המודל מתבצע בדומה למתואר בפרק הקודם בצורה self-supervised על חיזוי המילה הבאה, כאשר ה-backpropagation מתבצע אחורה דרך רצף הפרדיקציות (backpropagation through time – BPTT). BPTT נצרך, מכיוון שהפרדיקציה בזמן t תלויה ב- אשר תלוי ב- וכן הלאה, ולכן העדכון צריך להתבצע עבור המשקולות שבעזרתן כל ה- מחושבים. מכיוון שכל החישובים של ה-RNN גזירים, ניתן לבצע BP גם על פרדיקציות שהתבצעו בעבר באותו הרצף. האימון מתבצע על ידי* ***teacher forcing****, שמשמעותו היא שה-loss עבור מילה מתבססת על הרצף האמיתי, ולא על פרדיקציית המודל. שיטה זו מאפשרת למידה יציבה ומהירה יותר.*

RNN למטלות NLP נוספות

*קיימות מטלות נוספות ב-NLP אשר ניתן למדל בצורה קלה באמצעות RNN. מטלה אחת כזאת היא sequence labeling – חיזוי של class עבור כל מילה במשפט. דוגמה לבעיה ספציפית של sequence labeling היא part-of-speech tagging, תיוג חלק דיבר (כלומר שם עצם, פועל וכדו') עבור כל מילה במשפט. RNN היא ארכיטקטורה קלאסית לבעיה זו, מכיוון שהמודל מקבל רצף של מילים, יודע להתחשב במילים בסביבת המילה כדי להסיק את משמעותה, וכן ניתן באמצעותו להחזיר פרדיציה על כל מילה על ידי הוספת שכבה אשר מקבלת את וחוזה – התיוג המתאים.*

*בעיה נוספת היא sequence classification. בעיות ספציפיות של sequence classification הן למשל ניתוח רגשות (sentiment analysis) – האם ביקורת מסויימת על סרט היא חיובית או שלילית, Spam detection – האם מייל מסויים הוא spam או לא. בבעיות אלה הרשת נדרשת לעבור על כל הרצף ולתת תיוג. ניתן לבצע זאת על ידי RNN – העברת כל הרצף ברשת ושימוש בוקטור האחרון לביצוע הקלסיפיקציה (הוספה של שכבת קלסיפיקציה שמקבלת את ומסווגת את הרצף). במהלך האימון, הרשת תלמד לקודד לתוך את האינפורמציה הרלוונטית לסיווג הרצף.*

*עוד שימוש אפשרי ל-RNN הוא text generation – יצירה של טקסט. Text generation הוא חיוני בבעיות כגון question answering, תרגום, תמצות טקסט, תיקון שגיאות כתיב, ועוד. בבעיות אלה הטקסט שנוצר נדרש להיות נכון מבחינה תחבירית וטבעי לאדם הקורא אותו. ניתן לבצע זאת באמצעות שערוך ההסתברויות למילה הבאה בהינתן המילים הקודמות, ודגימה של המילה הבאה מתוך אותה ההתפלגות. לאחר שנבחרה המילה הבאה ניתן לשערך מחדש את פונקציית ההתפלגות למילה שאחריה ולחזור על התהליך עד לסיום. תהליך זה נקרא* ***autoregressive generation*** *או*  ***causal LM generation****.*

שיפורים ל-RNN

*ארכיטקטורת ה-RNN הינה מאוד פשוטה ובסיסית, המשקולות היחידות שנלמדות הן - ה-embedding של המילון, W,U המטריצות שמכפילות את ו-V שכבת המוצא. לכן, כוח הייצוג שלהם מוגבל, כלומר יהיה קשה לבצע איתן מטלות אשר אינן קרובות ללינאריות. קיימים מספר שיפורים לרשת הבסיסית. שיפור ראשון הוא stacked RNN. בשיטה זו, במקום להשתמש ברשת יחידה, משורשרות מספר רשתות RNN כך שכל רשת מקבלת ככניסה את המוצא של הרשת הקודמת. בשיטה כזאת ניתן לבצע עיבוד מורכב יותר, מכיוון שהמוצא עובר מספר שכבות (מספר רשתות).*

*בעיה נוספת הקיימת ב-RNN היא שהרשת בזמן t "ראתה" רק את העבר ולא את העתיד. עם זאת, ישנן משימות בהן המשפט כולו קיים, ולכן קיימת אינפורמציה שאנו לא משתמשים בה לצורך הפרדיקציה. פתרון אפשרי לבעיה הוא bidirectional RNN, רשת המורכבת משני RNNs נפרדים אשר אחד מקבל את הרצף מהתחלה לסוף ואחד מהסוף להתחלה. שני הייצוגים (וקטורי הזיכרון) משורשרים לוקטור אחד אשר נעשה בו שימוש לטובת הפרדיקציה:  
כאשר הם הייצוגים של הרשת הראשונה (forward) והשניה (backward) של המילה t. בצורה כזאת, מכיל אינפורמציה הן מהעבר והן מהעתיד.*

*בעיה נוספת ב-RNN הבסיסי הוא ההתמודדות עם רצפים ארוכים. מכיוון שהוקטור עובר טרנספורמציה כל נקודת זמן, קשה למידע להישאר בו לאורך רצפים ארוכים. לצורך העניין, אינפורמציה שקודדה בזמן בתוך ונדרשת בזמן , תצטרך לעבור פעמים את פונקציית הזהות (או פונקציה דומה), על מנת שיהיה אפשר להשתמש בה ב-. בעיה זו חריפה עוד יותר כאשר הרשת נדרשת בנוסף להשתמש ב- לחיזוי בזמן , למשל במקרה של sequence labeling. בעיה נוספת הקיימת כאשר הרצפים ארוכים היא שה-BPTT צריך לעבור שכבות רבות ולכן קיים חשש ל-vanishing gradients.אחד מהפתרונות לבעיה הוא ה-LSTM - long short-term memory. רשת זו לומדת "לשכוח" אינפורמציה לא רלוונטית, ולהוסיף מידע אשר יהיה נדרש בהסתברות גבוהה בעתיד. ישנם שני שינויים מרכזיים ב-LSTM, אחד הוא הוספה של "שערים" (gates). שערים אלה אחראים על "זיכרון" ו"שיכחה" של המודל. פרקטית, אלו שכבות מהצורה הבאה:  
הסיגמואיד מאלץ את המוצא של השכבה להיות בין אפס לאחד. כאשר מכפילים מוצא זה בוקטור אחר, כל האברים הקרובים לאפס בשער יתאפסו בווקטור האחר, וכל הוקטורים הקרובים לאחד בשער ישארו ללא שינוי בוקטור האחר. כלומר, חלק מהאינפורמציה "נשכח" וחלק ממנה נשאר. כל השערים הם מאותה הצורה:  
השינוי השני הוא הוספת "וקטור קונטקסט", אשר מועבר משלב לשלב בדומה ל-. וקטור זה עובר עיבוד (כלומר עובר שער) והתוצר של העיבוד הופך ל-. דבר זה מאפשר לבצע הבחנה בין וקטור ה"זיכרון" לוקטור שבו נעשה שימוש לפרדיקציה בזמן t.*

*כעת נצלול לארכיטקטורה. וקטור (הקונטקסט) מורכב מסכום של שני איברים:  
האיבר הוא הרכיב של שעובר מהשלב הקודם . כלומר, עובר ב"שער" (forget gate) אשר שולט באיזה אינפורמציה* ***נמחקת*** *מ-, על ידי הכפלה באפס. האיבר שולט באיזו אינפורמציה* ***מתווספת*** *ל-, כלומר מה אנחנו מעוניינים "לזכור". דבר זה מתבצע על ידי הכפלה של :  
בשער . היא האינפורמציה שמופקת בשלב t, ו- שולט באיזה חלק מהאינפורמציה אנחנו מעוניינים לאגור לשלבים הבאים. לאחר ה"מעבר בשער" (ההכפלה ב-), הוקטור מתווסף ל-. לאחר שמחושב ניתן לחשב את על ידי העברה בשער נוסף (על מנת להנגיש "החוצה" את האינפורמציה הרלוונטית מתוך ):*

*בצורה הזאת, ארכיטקטורת ה-LSTM מסוגלת להתבסס על וקטור זיכרון משוכלל יותר, וכך גם להתמודד בהצלחה עם רצפים ארוכים וגם לבצע את ההפרדה בין הזיכרון לפרדיקציה.*

Encoder-Decoder RNN

*בנוסף ל-sequence labeling, sequence classification ו-text generation, קיימת מטלות מסוג sequence to sequence. במטלות אלה הרשת מקבלת רצף של מילים ונדרשת להוציא רצף של מילים. הדוגמה הקלאסית לכך היא תרגום (machine translation), ובנוסף בסיכום (summarization), דיאלוג (כמו chatGPT) ו-question answering. ניתן לבצע מטלות אלה על ידי RNN על ידי שימוש ב-encoder ו-decoder. ה-encoder יקבל את רצף הכניסה ויקודד אותו לוקטור בגודל קבוע (למשל לתוך ). לאחר מכן ה-decoder יקבל את הקידוד כ-input וישתמש בו על מנת ליצור את רצף המוצא מילה אחרי מילה (בדומה ל-text generation שהזכרנו לעיל).*

Attention

*הבעיה המרכזית הקיימת בארכיטקטורת ה-encoder-decoder ב-RNN היא ה-information bottleneck – "צוואר הבקבוק". מכיוון שכל המידע מה-encoder מקודד לווקטור יחיד () באורך קבוע, כמות המידע שיכולה להיות מקודדת בו היא מוגבלת. עבור משפטים ארוכים, האינפורמציה המצויה בתחילת המשפט עלולה "להימחק" ולא להגיע על לווקטור הקונטקסט. מכיוון של-decoder יש גישה אך ורק לווקטור הקונטקסט, המידע שהוא יכול לפענח מוגבל. מנגנון ה-attention בא לפתור בעיה זו. הרעיון מאחוריו הוא שהמפענח יקבל בכל שלב וקטור* ***שונה*** *המורכב מסכום משוקלל של האינפורמציה מהמקודד בנקודות הזמן השונות. פונקציית ה-attention תבחר את המשקל עבור כל hidden state של המקודד כך שהמידע שיגיע למפענח בנקודת הזמן הנוכחית יורכב מהמידע הרלוונטי עבורו. חישוב ה-hidden state של המפענח בנקודת זמן תושפע מהפרדיקציה הקודמת, המצב הקודם וכן – וקטור האינפורמציה המשוקללת מהמקודד:  
כאשר מחושב בצורה הבאה. בשלב הראשון מתבצע חישוב של המשקלים של כל אחד מה-hidden states של המקודד, במילים אחרות, אילו חלקים מתוך דגימת ה-input רלוונטיים לפענוח בנקודת הזמן הזו. הצורה הפשוטה ביותר למדל משקל זה היא על ידי מכפלת וקטורים:  
נעשה חישוב של ה-score עבור כל אחד מ- ולאחר מכן מתבצע softmax על מנת לנרמל את הערכים:  
לאחר מכן מחושב באמצעות המישקול של :  
בצורה כזאת, מחושב ממוצע משוקלל של כל ה-hidden states של ה-encoder () והוא נלקח בחשבון בפענוח.*

Transformers

*ארכיטקטורה דומה אך אחרת היא ה-transformer או ה-self-attention. הארכיטקטורה ממפה רצף של וקטורים לרצף של ווקטורים באותו האורך: . ה-transformer מורכב משכבות transformer blocks שכל בלוק מורכב ממספר שכבות במבנה קבוע. שכבת ה-self-attention מאפשרת לרשת גישה לכל ה-inputs ללא צורך ב-RNN. כל בלוק כזה מקבל וקטורים כ-input ומוציא וקטורים.*

*בשכבת זו, ה-attention מוגדר בצורה הבאה:  
כאשר:  
כלומר הם טרנספורמציה לינארית של (שלוש טרנספורמציות שונות). הוא ה-****query****, כלומר הערך שה-token הנוכחי שאנו מעבדים "מתשאל" את שאר ה-tokens, ו- הם ה-****keys****, ערכים שמייצגים את שאר ה-tokens. לאחר אימון ה-transformer, אם ה-token ה- אינפורמטיבי לטובת הפרדיקציה בשלב ה-, הדמיון בין ה-query ל-key יהיה גדול, ו- יהיה גדול. הוא ה****ערך*** *עצמו, כלומר הערכים שאותם "נעביר הלאה". אם יהיה גבוה, אזי יהיה ייצוג משמעותי לווקטור בווקטור המשוקלל . הסימון בסכום הוא כאשר מאמנים מודל שחוזה את המילה הבאה, ולכן הוא יקבל אך ורק את ה-tokens הקודמים.*

*על מנת להימנע מערכים גבוהים במכפלה , וכתוצאה מכך אימון לא יציב, מתבצע נירמול של ה-score בממדי הוקטור:  
בנוסף, ניתן לבצע את כל החישובים הנ"ל בצורה מקבילית על כל ה-tokens. ניתן לייצג את כל הכניסות כמטריצה , ולבצע את החישובים באמצעות כפלי מטריצות:  
ולחשב את כל ה-scores באמצעות . לאחר מכן נרצה לבצע softmax על פני השורות של . עם זאת, אם נבצע softmax רגיל, יתקבלו ערכים גם "מהעתיד", כלומר עבור ה-token ה-, היינו רוצים כי כל הערכים הגדולים מ- יתאפסו בשורה ה-. לכן לפני ביצוע ה-softmax ניתן להציב במשולש העליון של המטריצה כך שה-softmax יתאפס במקומות הללו.*

*ה-transformer block מורכב מהשלבים הבאים. בשלב הראשון מתבצע ה-self-attention שתואר לעיל. לאחר שלב זה מתקבלים וקטורים . לאחר מכן מתבצעת נורמליזציה – layer normalization, אשר מנרמלת את הממוצע וסטיית התקן של כל הוקטורים (אותו הרעיון של batch normalization). הנורמליזציה מתבצעת עבור כל אחד מהווקטורים בנפרד. עבור כל token, מחושב הממוצע וסטיית התקן של הערכים שלו ומתבצע נרמול. בדומה ל-batch normalization, קיימים שני פרמטרים נלמדים אשר שולטים על הממוצע וסטיית התקן. לסיכום, עבור כל וקטור מתבצעים השלבים הבאים:  
לאחר מכן, כל וקטור (מנורמל) עובר בשכבת FC ולאחר מכן בעוד נירמול. מתבצע גם residual connection אשר "עוקף" את ה-self-attention ואת ה-fully connected (אבל לא את הנורמליזציה). לסיכום, משוואת ה-attention block:  
כאשר היא שכבת fully connected (feed forward).*

*שיפור שנהוג לבצע לשכבת self attention הוא ה-multi head attention. שכבה זו מכילה N שכבות attention שפועלות במקביל, עם סט פרמטרים שונה לכל שכבה, כאשר בסופה של השכבה מתבצע שרשור (concatenation) של כל ה-outputs של השכבה והטלה לינארית (linear projection) חזרה לממד . הרציונל במספר שכבות attention הפועלות במקביל הוא שכל שכבה תתמקד בפן אחר של קשרים בין inputs, ויחד שכבות אלו יוכלו לתפוס את היחס השלם שבין ה-tokens.*

*שיפור נוסף שנהוג לבצע במשימות NLP הוא הוספת ה-positional encoding. הבעיה שה-positional encoding בא לפתור היא שבשונה מ-RNN, ה-transformers הוא אינווריאנטי לסדר, כלומר אין שום דרך למודל לדעת באיזה מקום כל token נמצא במשפט. אך בשפה טבעית המיקום של המילים הוא קריטי להבנת המשפט, ולכן הרשת מוכרחת לקבל בנוסף מידע לגבי מיקום ה-token במשפט. אחד מהפתרונות הוא הוספה של שכבת embeddings שתקודד את המיקומים. כלומר, הרשת תלמד ייצוג שונה עבור כל מיקום, וייצוג זה יתווסף לייצוג ה-embedding של המילה עצמה. למשל המשפט “The dog eat meat” יקודד ל:  
בצורה כזאת, המודל יקבל את האינפורמציה לגבי המיקום במשפט, למרות שהארכיטקטורה אינווריאנטית לסדר.*

*שיעור 6 – נדרש להשלים*

Semantic Role Labeling (SRL)

*תיוג תפקידים סמנטיים (semantic role labeling או SRL), הוא אחד מהדרכים המרכזיות למידול שפה. המטרה ב-SRL היא תיוג ה****תפקיד הסמנטי*** *(semantic role) של העצמים במשפט. ההנחה היא שמשפט בנוי* ***מנשוא*** *(predicate), (בדרך כלל הפועל של המשפט) ומחלקים, או* ***ארגומנטים****, שמתייחסים אליו. לכל ארגומנט ישנו תפקיד (role) התלוי במשפט. למשל במשפט: The child broke the window, תפקיד הילד הוא "השובר", ובמשפט Dana placed a book on the shelf, תפקידה של דנה הוא "המניחה". המשותף בשני המשפטים הוא שבעל התפקיד* ***גורם*** *לדבר לקרות* ***באופן רצוני****. במקרים כאלה, בעל התפקיד נקרא* ***סוכן*** *(agent). לעומת זאת, תפקיד החלון ("הנשבר") והספר ("המונח") הם תפקידים של חפצים דוממים, והם סבילים, כלומר קיימת* ***פעולה*** *שנעשית עליהם. תפקיד זה נקרא* ***תמה*** *(theme). תפקיד שלישי הוא תפקידו של המדף עליו הניחה דנה את הספר. תפקיד זה נקרא ה****מטרה*** *(goal). קבוצת כל התפקידים שאותם הזכרנו נקראת "Thematic roles", והיא מכילה כמה עשרות תפקידים. לדוגמה:*

* ***סוכן*** *(agent) – דבר רצוני הגורם לאירוע.*
* ***חווה*** *(experiencer) – מי שחווה אירוע.*
* ***כוח*** *(force) – דבר שאינו רצוני הגורם לאירוע.*
* ***תמה*** *(theme) – המשתתף אשר מושפע בצורה הגדולה ביותר מהאירוע (למשל הספר או החלון בדוגמאות לעיל).*
* ***תוצאה*** *(result) – התוצר הסופי של האירוע (event).*
* ***תוכן*** *(content) – למשל "הוא סיפר לי* ***בדיחה****".*
* ***כלי*** *(instrument) – כלי שנעשה בו שימוש באירוע.*
* ***"מרוויח"*** *(beneficiary) – מי שלמענו נעשה משהו, למשל "היא אפתה עוגה ל****אחותה****".*
* ***מקור*** *(source) – המקור של אובייקט מסוים כאשר קיים מאורע של העברה, למשל "הוא הסיר את הספר מה****מדף****".*
* *מטרה (goal) – המיקום הסופי של אובייקט כאשר קיים מאורע של העברה, למשל "הוא הניח את הספר על ה****שולחן****".*

*מידול של השפה בצורה כזו (באמצעות חלוקה לתפקידים סמנטיים) הוא אחד מהמידולים העתיקים ביותר. הוא נוסח בצורה מודרנית ב-1968 (Fillmore) וב-1985 (Gruber). קיימות כמה עשרות קטגוריות, אם כי לעיתים כאשר מבצעים את מטלת התיוג ב-NLU נעשה שימוש בתת קבוצה של קטגוריות. ניתן להשתמש ב-SRL בתור ייצוג בסיס של השפה אשר יהיה ברמה מסוימת אינווריאנטי לניסוח עצמו (לתחביר - syntax). כלומר, שני משפטים בעלי ניסוחים שונים אך עם אותם חלקי SRL יקבלו את אותו הייצוג. למשל, אם נתבונן בשני המשפטים הבאים:*

* Bill gave the book to Jane.
* Bill gave Jane the book.

*סדר המילים משתנה, אך המשמעות זהה. אם נשתמש בתפקידים סמנטיים למידול המשפט נקבל ייצוג זהה. תכונה זו שימושית למשל ל-machine translation.*

*עם זאת, ישנם אתגרים בשימוש בתפקידים סמנטיים. למשל, נדרש להגדיר סט של תפקידים סמנטיים שיהיה סטנדרטי קונסיסטנטי ומקובל. בנוסף, הגדרה פורמלית של תפקידים עלולה להיות קשה, שכן ישנם תפקידים שונים אשר קשה להגדירם בצורה מדויקת, וכן לפעמים נרצה להכליל את התפקידים בצורה יותר אבסטרקטית. הפתרון לבעיות אלה הוא להכליל מ-role (תפקיד) ל-proto-role, או במילים אחרות – generalized semantic roles.*

קורפוסים ושיטות תיוג

Proposition Bank (Palmer et al., 2005)

*המאגר proposition bank, או לחלופין PropBank, הוא שם של קורפוס (dataset) המכיל משפטים מתויגים לקלסיפיקציה של תפקידים סמנטיים. בקורפוס זה משתמשים גם ב-proto-roles וגם בתפקידים סמנטיים שספציפיים לפועל מסוים (verb specific semantic role). המאגר מורכב מכל הפעלים באנגלית, ועבור כל פועל – כל המשמעויות שלו. עבור כל משמעות של כל פועל מופיע משפט, כאשר מצוין במשפט מהו ה-proto-agent וה-proto-patient. תפקידים אלה מתויגים כ-Arg0 (proto-agent) ו-Arg1 (proto-patient). בנוסף, התפקידים Arg2, Arg3, Arg4 גם מוגדרים, אך הם פחות קשיחים מ-Arg0,Arg1, והם מקבלים משמעות שונה מעט עבור כל פועל. משמעויות אלה מוגדרות בתוך הקורפוס. למשל, Arg2 יכול לקבל את אחד התפקידים הבאים: Benefactive/ Instrument/ Attribute/ End State. Arg3 יכול לקבל את אחד התפקידים הבאים: Start Point/ Benefactive/ Instrument/ Attribute. Arg4 הוא בדרך כלל End Point. בנוסף, ישנם תפקידים שאינם ממוספרים, למשל זמן (TMP), מקום (LOC) ועוד. להלן שתי דוגמאות לפעלים המופיעים ב-PropBank:*

* ***Agree.01*** *(הפועל agree במשמעותו הראשונה)*
  + *Arg0: Agreer (זהו ה-proto-agent עבור פועל זה ספציפית)*
  + *Arg1: Proposition (זהו ה-proto-patient עבור פועל זה ספציפית)*
  + *Arg2: Other entity agreeing*
  + *דוגמה ראשונה:* [Arg0 The group] **agreed** [Arg1 it wouldn’t make an offer].
  + *דוגמה שניה:* [ArgM-TMP Usually] [Arg0 John] **agrees** [Arg2 with Mary] [Arg1 on everything]
* ***Fall.01***
  + Arg1: Logical subject, patient, thing falling (במקרה זה אין Arg0, שכן אין מישהו שמבצע את הפעולה)
  + Arg2: Extent, amount fallen
  + Arg3: start point
  + Arg4: end point, end state of arg1
  + *דוגמה ראשונה:* [Arg1 Sales] **fell** [Arg4 to $25 million] [Arg3 from $27 million]
  + דוגמה שניה: [Arg1 The average junk bond] **fell** [Arg2 by 4.2%]

*לסיכום, היתרונות במידול PropBank הם שבהינתן פועל ניתן בצורה קלה לתייג כל חלק מהמשפט לתפקיד המתאים, וכן שהתיוג קונסיסטנטי לכל הפעלים (Arg0, Arg1 וכו'). החיסרון העיקרי הוא ששיטה זו לא גנרית, אלא נדרשת הגדרה לכל פועל, וקשה לבצע הכללה אוטומטית לפעלים נוספים.*

NomBank (Meyers et al., 2004)

*מאגר זה, בשונה מ-PropBank, מתמקד בשמות עצם ולא בפעלים. למשל שני המשפטים הבאים אינם מכילים פועל כלל:*

* *The construction of the metro*
* *The agreement of Apple with IBM*

*להלן דוגמה מתוך המאמר המתאר את המאגר:*

* *Her gift of a book to John [NOM]*
  + *REL = gift*
  + *ARG0 = her*
  + *ARG1 = a book*
  + *ARG2 = to John*

*מאוחר יותר, נעשתה עבודה להוספת משפטים שמניים (nominal predicates – משפטים ללא פועל) ל-PropBank. למשל, בוצע תיוג של dataset נוסף – OntoNotes – המכיל משפטים כאלה.*

FrameNet (Baker et al., 1998)

*מטרת ה-database היא ניתוח של שפה בצורה אבסטרקטית יותר. ה-dataset מכיל "מסגרות" (frames), כלומר משפטים, בהם קיים* ***אירוע (event)*** *או* ***scenario*** *והמשתתפים באירוע. למשל, אירוע של “Commerce\_buy” יכיל קונה, מוכר, סחורה, כסף וכו'. גישה זו שונה מהגישות השונות בהן כל פועל מודל בנפרד, במידול זה ממודלות מסגרות אשר יכולות להכיל מילים שונות בעלות משמעות זהה. למשל, שני המשפטים הבאים:*

*"John* ***purchased*** *a book from the bookstore."  
"Mary* ***bought*** *a phone on the shopping website."*

*PropBank יתן תיוג לכל פועל בנפרד, אך FrameNet יסווג את שני המשפטים לאותה מסגרת מכיוון ש- purchased ו- bought הן מילים נרדפות בעלות משמעות זהה (שתיהן יסווגו כ-Commerce\_buy).*

*FrameNet מקשר את המילים למסגרת אליה הם שייכים. עבור כל מסגרת, קיים תיוג של* ***הגדרת המסגרת****, כלומר איזה סוג אירוע או scenario קיים במסגרת, איזה מילים משמשות כ"****trigger****", כלומר אילו מילים גורמות לכך שהאירוע יתויג בצורה מסוימת, וכן מה* ***התפקידים של הישויות השונות*** *שבמסגרת.*

*קיימים מספר יתרונות במידול כזה. ראשית, כל תפקיד (role) מוגדר בצורה מאוד מדויקת (מוכר, קונה וכו'). לעומת זאת, בשיטות אחרות קיימים מקרים בעייתיים. למשל, ב-PropBank, Arg0 בדרך כלל מתאר את מי שמבצע את הפעולה, כמו במקרה הבא:*

*"[John Arg0] broke the window".*

*עם זאת, ישנם מקרים יוצאי דופן, למשל:*

*“[The window Arg0] broke".*

*ניתן לראות שבמשפט הראשון Arg0 היה* ***השובר****, ואילו במשפט השני –* ***מה שנשבר****. ב-FrameNet התיוג קונסיסטנטי – בשניהם התיוג שניתן ל-“the window” יהיה "הדבר שנשבר". יתרון נוסף הוא שב-FrameNet קיימת הכללה בין מספר פעלים בעלי משמעות דומה כפי שהודגם לעיל (purchased, bought). מצד שני, מכיוון שהשיטה יותר high level, ישנם פחות תיוגים של פעלים שונים ומגוונים ומשמעויות שונות של פועל. בנוסף, מכיוון שהתיוג של התפקידים מאוד מוגדר ומפורט (fine-grained), התיוגים יהיו דלילים (פחות תיוגים לכל תפקיד) ולכן יהיה יותר קשה להכליל.*

Semantic Role Labeling (SRL)

*מטרת ה-SRL היא תיוג אוטומטי של כל התפקידים הסמנטיים של כל הארגומנטים המתייחסים לנשוא (predicate). כפי שתואר לעיל, אנו מתייחסים לנשוא (למשל הפועל כאשר קיים פועל במשפט – משפט פעלי ולא שמני) בתור החלק המרכזי במשפט, וכל חלקי המשפט (הארגומנטים) מתייחסים אליו בצורות שונות. המטרה ב-SRL היא לסווג בצורה אוטומטית את התפקידים של הארגומנטים, כלומר באיזה אופן הם מתייחסים לנשוא במשפט. כשמתייחסים ל-SRL בדרך כלל מתייחסים גם לבעיית* ***מציאת הארגומנטים*** *(בנוסף לסיווגם), כלומר לזהות אילו מילים מרכיבות ארגומנט, וכן לעיתים* ***זיהוי הנשוא*** *בצורה אוטומטית. כפי שראינו לעיל ב-PropBank כל פועל יכול לקבל מספר מובנים, ולכן ניתן לבצע* ***זיהוי סוג הנשוא****, או המובן, עבור פועל מסוים.*

תחביר

*הדרך הבסיסית ביותר לתיאור תחביר המשפט היא באמצעות חלקי דיבר (שם עצם, פועל וכו').*

*צירופים (Syntactic Phrases/Constituents) הם קבוצת מילים המהוות יחידה תחבירית אחת. ישנם צירופים שמניים (Noun phrases – NP), למשל: I saw [a dog] או I saw [a small dog with a black tail], המילים המוקפות בסוגריים הם יחידה עצמאית שמנית – ללא פועל. ישנם צירופים פעליים (Verb phrases – VP), למשל I [walk] או I [walk home quickly but surly] – בשתי הדוגמאות הנ"ל כל המילים המופיעות בסוגריים מתייחסות לפועל או למילה אחרת המתייחסת לפועל. יש צירופים תאריים (Adjective phrases – AP) – אשר כל המילים בתוך הצירוף מתייחסות לתואר, למשל The table [made of solid oak] can seat eight people. בנוסף, יש צירוף יחס (Prepositional Phrases (PP)) שהגרעין שלו הוא מילת יחס, למשל I’m taking the train [from Tel-Aviv].*

*כאשר בונים את הצירופים נוצרת היררכיה של צירופים, למשל: [[birds [that swim]] fly] או [I [like [old books]]]. לכן, ניתן בצורה כזאת לייצג את מבנה המשפט בצורת עץ. עץ זה נקרא עץ תחביר, או Syntactic Tree. העלים בעץ זה הם המילים. למשל:*

*בדוגמה זאת, כל העלים (המילים) שמתחת ל-VP יוצרים צירוף פעלי, כל העלים שמתחת ל-NP יוצרים צירוף שמני וכן הלאה.*

*אם נחזור לבעיית ה-SRL, ניתן להשתמש בעץ התחביר לביצוע SRL. ניתן לתייג כל צירוף לתפקיד סמנטי ובצורה כזאת לתייג את כל המשפט. זו הגישה שהייתה נפוצה בעבר, אך יש לה חסרונות, למשל קושי בהתמודדות עם משפט שמני (ללא פועל). כיום רשתות נוירונים מאפשרות גישה ישירה יותר לבעיה אשר אינה מצריכה בניית עץ.*

SRL באמצעות רשתות

*ניתן להתייחס ל-SRL כבעיית קלסיפיקציה של tokens. ניתן להשתמש בארכיטקטורת BERT לטובת המשימה, שכן היא מאפשרת פרדיקציה לכל טוקן. הפרדיגמה הנפוצה באופן כללי במטלות NLP (לפחות עד עידן ה-LLMs) היא ביצוע pre-training של הרשת על משימה self-supervised – למשל חיזוי טוקנים חסרים (MLM) ב-encoders, או השלמת המילה הבאה (NSP) ב-decoders, ולאחר מכן ביצוע fine-tuning על המשימה הספציפית. היתרון הגדול בגישה זו הוא שאת ה-pre-training ניתן לבצע באמצעות כמויות גדולות מאוד של data שאינו מתויג, וכך את ה-fine-tuning ניתן לבצע באמצעות כמות קטנה יחסית של data מתויג למשימה הספציפית. ייתרון נוסף הוא שניתן לעשות fine-tuning למשימות רבות על בסיס רשת יחידה שעברה pre-training.*

*אחד האתגרים בשימוש ב-BERT ל-SRL הוא שהטוקניזציה לא מתבצעת לפי מילים אלא לפי חלקי מילים, ואילו ב-SRL התיוג מתבצע לכל מילה ואף ל-span של מספר מילים. אחת מהגישות לפתרון הבעיה היא BIO. בגישת ה-BIO (Beginning, Inside, Outside) מטלת ה-SRL מומרת למשימת קלסיפיקציה עבור כל token ל- קטגוריות, כאשר הוא מספר הארגומנטים האפשריים. עבור כל ארגומנט קיים class של B – beginning, של I – inside, ובנוסף, קיימת קטגוריה ל-token שאינו שייך לאחד הארגומנטים – O שמסמן outside. למשל, אם קיים רק Arg0 יהיו שלושה classes: B-Arg0, I-Arg0 ו-O.*

***בזמן האימון****, הרשת תחזה כל חלק-מילה (כל טוקן) לפי התיוג המתאים (למשל B-Arg0, I-Arg0, O).* ***בזמן ה-inference*** *המודל נדרש להחזיר סיווג לכל מילה (ולא לכל token). על מנת לעמוד בדרישה, ניתן בשלב ראשון לבצע פרדיקציה אך ורק ל-tokens המשמשים כתחילת מילה, וכך לקבל את התיוג לתחילות מילה. בשלב הבא, כל ה-tokens שבהמשך לראשון (B) שקיבלו פרדיקציה זהה לזו של הראשון מצטרפים לראשון. עם זאת, SRL הוא בעיה מסוג span-based-application, כלומר זהו לא תיוג לכל מילה (או טוקן) ומצד שני לא תיוג לכל המשפט, לכן זהו לא פתרון טבעי לבעיה.*

*כעת נגדיר בצורה פורמלית יותר את ה-span-based-application. Span הוא רצף של טוקנים. בהינתן טקסט המורכב מהטוקנים: , span הוא סדרה רציפה של טוקנים המתחילה בטוקן i ומסתיימת בטוקן j, כאשר . לעיתים ישנו אילוץ על אורך הרצף, כלומר . נגדיר כאוסף כל ה-spans החוקיים במשפט .*

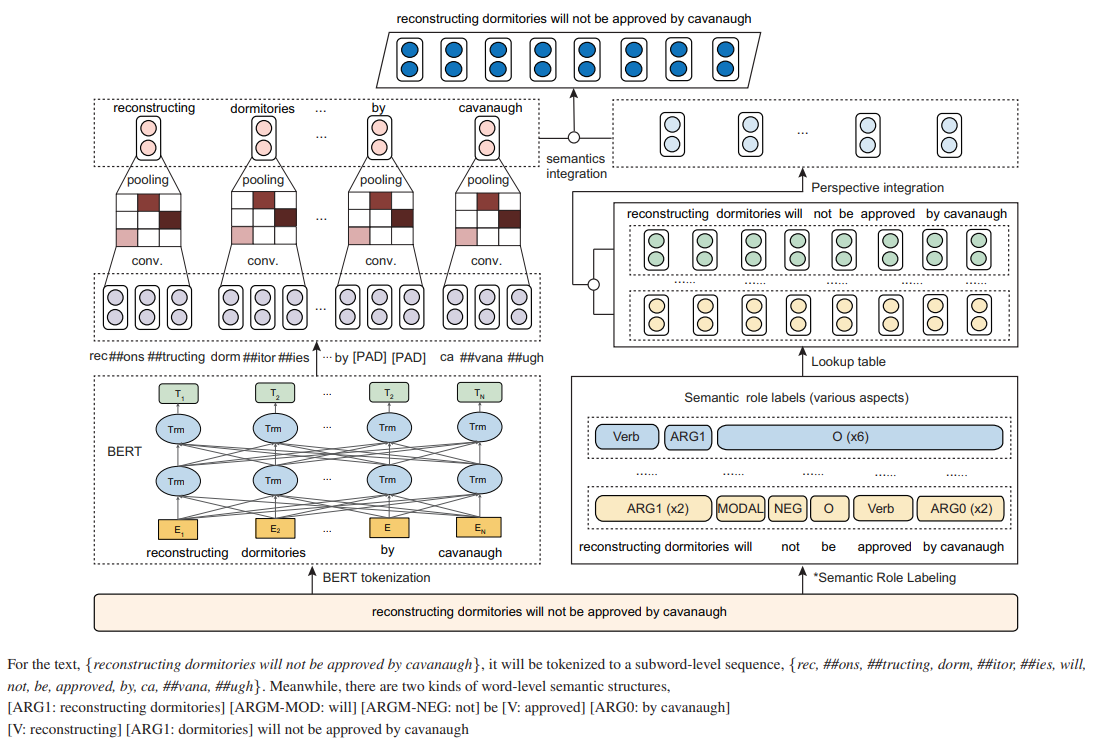
*אנו מעוניינים להשתמש בייצוג שנלמד באמצעות BERT על מנת ללמוד שני דברים: ייצוג לגבולות ה-span וייצוג של התוכן של ה-span. על מנת ללמוד את ייצוג התוכן של span ניתן למצע את כל הייצוגים של BERT של כל הטוקנים שבתוך ה-span:  
כאשר הוא הייצוג של BERT עבור טוקן . בנוסף, ניתן לבצע שרשור של על מנת לקבל ייצוג של הגבולות של ה-span. חיסרון מרכזי בשיטה זו הוא שב- אנו ממצעים על פני הייצוגים ולכן מאבדים את המידע לגבי מיקומי הטוקנים שעליהם ממצעים. בנוסף, מכיוון שמבצעים ממוצע, כל הטוקנים מקבלים משקל שווה, אף על פי שייתכן שטוקנים מסויימים יותר משמעותיים למשימה מאשר טוקנים אחרים.*

*על מנת להתגבר על הבעיות הללו, במקום למצע ניתן להשתמש בשכבת self-attention על הטוקנים ב-span. כלומר:  
ובנוסף, ניתן להשתמש בשכבת feed forward לייצוגי ההתחלה והסוף: , ולהפיק ייצוג של ה-span באמצעות: . לאחר שיש ייצוג של span, ניתן להתייחס לבעיית ה-SRL כבעיית קלסיפיקציה:*

*האבלואציה של אלגוריתם ה-SRL תתבצע באמצעות בדיקה עבור כל ארגומט האם הוא מתפרש בדיוק על הטוקנים אותם האלגוריתם חוזה. לכן, עבור כל ארגומנט קיים ציון של אפס או אחד. לאחר מכן ניתן להשתמש במטריקות סטנדרטיות כגון F1, recall, precision.*

SemBERT

*עבודה זו פורסמה בשנת 2020, ומטרתה* ***שיפור מטלות NLP באמצעות הוספת מידע של SRL****. הרעיון הוא שאם מודל שפה (למשל BERT) יקבל בנוסף למשפט את ה-SRL שלו זה עשוי לשפר את ביצועי המודל על מטלות שונות בעיבוד שפה. המאמר משתמש ב-BERT מאומן, ומוסיף למידה של ייצוג (embedding) התפקידים הסמנטיים בנוסף לייצוג הטוקנים של המילון. המאמר מראה שכאשר מבוצעת אינטגרציה של ה-SRL עם מודל שפה סטנדרטי, הביצועים של המודל משתפרים (למשל בדקו על משימות קלסיפיקציה מה-benchmark של GLUE, על Textual Entailment ועל Semantic Similarity Tasks). להלן איור הלקוח מתוך המאמר המתאר את השיטה:*

**

*בחלק השמאלי של האיור, המשפט עובר ברשת BERT סטנדרטית, ולאחר מכן שכבת קונבולוציה ו-pooling על מנת לצמצם את הייצוג של המשפט למילים (במקום טוקנים). בצד ימין ניתן לראות כי המשפט עובר במודל (או מספר מודלים) של SRL ומתקבל ייצוג לכל מילה. הייצוגים עוברים שכבה לינארית ושרשור לייצוג של המילים שחולצו מ-BERT.*

מטלות שונות

*ישנן מטלות (tasks) שונות אשר נושקות ל-SRL. למשל –* ***Implicit Argument Detection****, זיהוי ארגומנט אשר לא מופיע בצורה מפורשת במשפט. למשל בדוגמה הבאה:*

*This house has a new owner. The sale was finalized 2 days ago.*

*במשפט השני, הנשוא הוא was finalized, ה-Arg1 הוא The sale, אבל ה-Arg0 (הסוכן, "מי סיים את המכירה?") לא מופיע במשפט. במקרים כאלה, נרצה למצוא את הארגומנטים הנסתרים במשפט, אשר בדרך כלל מופיעים במשפט אחר ולא במשפט שהפועל (או הנשוא) מופיע.*

*דוגמה נוספת למטלה הקרובה ל-SRL היא* ***Connotation frames****. במטלה זו, אנו מעוניינים לזהות את הקונוטציה (למשל קונוטציה חיובית או שלילית) שיש לחלקים במשפט. זוהי מטלה יותר "עדינה" מזיהוי משמעות סמנטית "יבשה" של חלקי המשפט. למשל, במשפט הבא:*

*He survived the bombing.*

*הקונוטציה של תחילת המשפט (He survived) היא חיובית, כלומר מייצרת רגש של אמפטיה אצל הקורא, ואילו סוף המשפט (the bombing) מייצרת קונוטציה שלילית, או תחושה שלילית אצל הקורא.*

חילוץ מאורעות

*חילוץ מאורעות (event extraction) יכול להיחשב כתת תחום של SRL. מטלה זו היא מסוג חילוץ מידע (information extraction), ונועדה לחלץ מתוך טקסט, שהוא data לא מובנה (unstructured), מאורעות. לאחר שחולץ מאורע, ניתן להתייחס אליו כ-data מובנה, למשל, ניתן עבור כל מאורע לומר מהו סוג המאורע, מיקום וזמן של המאורע, והמשתתפים במאורע.*

*חילוץ מאורעות מורכב בדרך כלל מארבע תתי בעיות. לצורך ההדגמה נשתמש במשפט "* China purchased two nuclear submarines from Russia last month"*:*

1. *זיהוי ה-trigger (זיהוי המאורע) – trigger identification (TI): זיהוי הפועל או שם העצם שמעיד על המאורע. בדוגמה שלעיל, ה-trigger הוא "purchased" שכן מילה זו מעידה על מאורע הרכישה שהתבצע.*
2. *סיווג של ה-trigger (סיווג המאורע) – trigger classification (TC): לאחר שבשלב הקודם זיהינו מהו ה-trigger, כעת ניתן לסווג אותו לסוגי מאורעות. זוהי משימת קלסיפיקציה מתוך סט סופי של סוגי מאורעות. בדוגמה שלעיל, הסיווג של purchased הוא TRANSFER-OWNERSHIP.*
3. *זיהוי הארגומנטים – argument identification (AI): זיהוי (identification) של כל הארגומנטים המשוייכים ל-trigger. למשל בדוגמה לעיל הארגומנטים יכולים להיות: China, two nuclear submarines, Russia, last month.*
4. *סיווג הארגומנטים (argument role classification) – argument classification (AC): סיווג כל אחד מהארגומנטים שנמצאו בשלב הקודם לאחד מהסוגים שהוגדרו מראש ולפי סוג ה-trigger. למשל, בדוגמה שלעיל, China היא הקונה (BAYER), two nuclear submarines הם ה-artifact (הדבר שמוכרים), Russia היא המוכרת (SELLER) וlast month הוא הזמן (TIME).*

*כאמור, סיווג המאורע (ה-trigger) הוא מתוך רשימת מאורעות המוגדרת מראש, ועבור כל מאורע ישנה רשימה סגורה של ארגומנטים חוקיים. ניקח לדוגמה את ה-dataset ACE 2005. ב-dataset זה מתוייגים שמונה משפחות של סוגי מאורעות:*

* LIFE
* MOVEMENT
* TRANSACTION
* BUSINESS
* CONFLICT
* CONTACT
* PERSONELL
* JUSTICE

*ועבור כל משפחת מאורעות ישנם תתי מאורעות מתאימים. למשל, עבור המאורע LIFE קיימים 33 תתי מאורעות:*

* BE-BORN
* MARRY
* DIVORC
* INJURE
* DIE
* Etc.

*ב-ACE 2005 יש 28 סוגי ארגומנטים. חלקם ספציפיים למאורע מסוים (למשל VEHICLE-ARG למאורע MOVEMENT.TRANSPORT) חלקם מתאימים למספר מאורעות (למשל PERSON-ARG) וחלקם מתאימים לכל סוגי המאורעות (PLACE-ARG, TIME-ARG).*

*לצורך ביצוע חילוץ מאורעות ניתן להשתמש ב-BERT בגישת sequence labeling, כאשר כל token יסווג כאחד מהארגומנטים (מלבד ה-trigger). ניתן גם להתייחס לבעיה כ-span-based classification ולהשתמש ב-BIO format כפי שהוצג לעיל עבור SRL. בנוסף, עבור שתי הגישות, ניתן לבצע את ארבעת המשימות (TI,TC,AI,AC) בצורה טורית (pipeline), משימה אחרי משימה, או בבת אחד – רשת אחת לזיהוי וסיווג של ה-trigger והארגומנטים.*

*מבחינת האבלואציה, נהוג לבצע על כל תת משימה (ארבעת תתי המשימות שלעיל) בנפרד. על כל אחת מהמשימות ניתן להשתמש במדדים סטנדרטיים של recall, precision ו-F1. להלן טבלה המסכמת את נוסחאות המטריקות:*

|  |  |  |
| --- | --- | --- |
| ***Formula*** | ***Metric*** | ***Task*** |
|  | *Recall* | *Trigger Identification (TI)* |
|  | *Precision* |
|  | *F1* |
|  | *Recall* | *Trigger Classification (TC)* |
|  | *Precision* |
|  | *F1* |
|  | *Recall* | *Argument Identification (AI)* |
|  | *Precision* |
|  | *F1* |
|  | *Recall* | *Argument Classification (AC)* |
|  | *Precision* |
|  | *F1* |

Named Entity Recognition

*Named Entity Recognition, או בקיצור NER, היא מטלה נוספת בניתוח שפה. המטרה של NER היא זיהוי וסיווג ישויות (named entities) בתוך הטקסט. למשל, במשפט הבא:*

*“Apple Inc. is planning to open a new store in New York City”*

*נרצה לזהות את Apple Inc. כ****ארגון*** *(organization) ואת New York Cite כ****מקום*** *(location). להלן דוגמאות לישויות שונות:*

* *Person ("John Smith" or "Alice")*
* *Organization ("Google" or "United Nations")*
* *Location ("New York City" or "Mount Everest")*
* *Date ("January 1, 2023" or "Monday")*
* *Time ("2:30 PM" or "noon")*
* *Money ("$100" or "5 euros")*
* *Percentage ("20%" or "half")*
* *Etc.*

*ביצוע NER נעשה בדרך כלל בשיטות span-based כפי שראינו לעיל ב-SRL.*

Relation Extraction

*Relation Extraction הוא מטלה נוספת בהבנת שפה, אשר מטרתה זיהוי הקשרים שבין ישויות שונות בטקסט. את הישויות ניתן לזהות באמצעות NER ו-Relation Extraction יכול להתבסס על הישויות שזוהו, ולסווג את הקשרים בין הישויות. לדוגמה, במשפט “Barack Obama was born in Hawaii” הישויות הן Barack Obama (מסוג Person) ו-Hawaii (מסוג Place), והקשר ביניהם הוא born\_in. במשפט “John sent $100 to Mary”, הישויות הן John, 100$ ו-Mary, והקשר ביניהם הוא שליחה (sent).*

*הבסיס של Relation Extraction הוא ה-domain, ה-classes וה-relations. ה-domain מגדיר מהו ה-scope שעליו עובדים. למשל, domain יכול להיות רפואה, ספורט וכדו'. בתוך ה-domain מוגדרים ה-classes, שהם הקטגוריות של הישויות שקיימות ב-domain. למשל, תחת ה-domain של* ***רפואה*** *יכולות להיות הקטגוריות הבאות:*

* ***Diseases****: "diabetes," "cancer," "hypertension," or "influenza."*
* ***Medications****: "aspirin," "antibiotics," "insulin," or "chemotherapy drugs."*
* ***Symptoms****: "fever," "cough," "headache," or "fatigue."*
* *Etc.*

*על ה-classes שבתוך ה-domain מוגדרים* ***קשרים*** *(****relations****). למשל, תחת ה-domain של רפואה ניתן להגדיר את הקשרים הבאים:*

* ***Treats****: "[ Medications aspirin] treats [Diseases headache]" or "[ Medications chemotherapy drugs] treat [Diseases cancer]."*
* ***Causes****: "[Factor smoking] causes [Diseases lung cancer]" or "[Factor unhealthy diet] causes [Diseases obesity]."*
* ***Diagnoses****: "[Medical Professionals Dr. Smith] diagnosed [Patient the patient] with [Diseases diabetes]."*
* *Etc.*

*נביא דוגמה נוספת. להלן קטע מתוך טקסט:*

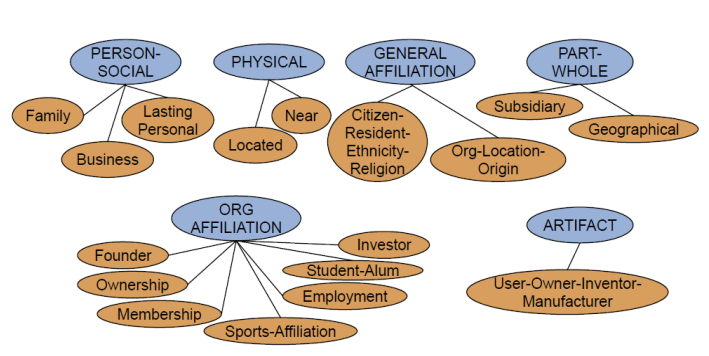
Citing high fuel prices, **[ORG United Airlines]** said **[TIME Friday]** it has increased fares by **[MONEY $6]** per round trip on flights to some cities also served by lower-cost carriers. **[ORG American Airlines]**, a unit of **[ORG AMR Corp.]**, immediately matched the move, spokesman **[PER Tim Wagner]** said. **[ORG United]**, a unit of **[ORG UAL Corp.]**, said the increase took effect **[TIME Thursday]** and applies to most routes where it competes against discount carriers, such as **[LOC Chicago]** to **[LOC Dallas]** and **[LOC Denver]** to **[LOC San Francisco]**.

|  |  |  |
| --- | --- | --- |
| ***Entity*** | ***ID*** | ***Class*** |
| *United* |  | *Organization* |
| *UAL* |  | *Organization* |
| *American Airlines* |  | *Organization* |
| *AMR* |  | *Organization* |
| *Tim Wagner* |  | *Person* |
| *Chicago* |  | *Location* |
| *Dallas* |  | *Location* |
| *Denver* |  | *Location* |
| *San Francisco* |  | *Location* |

*בטקסט זה, ה-domain הוא תעופה, והוא מכיל את:*

*וכן קיימים הקשרים הבאים:*

*כלומר (United ו-UAL) מקיימים קשר של PartOf. ב-dataset של ACE מתויגים 17 סוגי קשרים שונים:*

**

*למשל:*

* *[PER He] was in [GPE Tennessee]:* ***Physical-Located***
* *[ORG XYZ], the parent company of [ORG ABC]:* ***Part-Whole-Subsidiary***
* *[PER Yoko]’s husband [PER Jhon]:* ***Person-Social-Family***
* *[PER Steve Jobs], co-founder of [ORG Apple]:* ***Org-AFF-Founder***

ניתן לגשת לבעיית ה-Relation Extraction כבעיית סיווג של sequence. BERT יקבל את המשפט בכניסה, ויחזה במוצא של token ה-[CLS] שלו את סוג הקשר המופיע במשפט. ניתן להשתמש בייצוג Partly de-lexicalized, כלומר להחליף חלק מהמילים במשפט ב-placeholders, במקרה שלנו להחליף את ה-Entities שעליהם מעוניינים לחזות את הקשר ביניהם ב-class המתאים. למשל, במקום לתת ל-BERT את המשפט He was in Tennessee, ניתן לו את המשפט [PER] was in [GPE]. שיטה זו מאפשרת להכליל בצורה יותר טובה ולהימנע מ-overfitting.

Coreference

*Coreference היא מטלה נוספת בהבנת שפה טבעית. המטרה היא להבין האם שתי ישויות המוזכרות (Entity coreference) או שני האירועים המוזכרים (Event coreference) מתייחסים לאותו הדבר. למשל במשפט הבא:*

***Barack Obama*** *started his speech. It was the first time the* ***former president of the United Stated*** *visited that city.*

*האם* ***Barack Obama*** *ו-* ***former president of the United Stated****מתייחסים לאותה ישות (אותו אדם), והאם בדוגמה הבאה:*

*s1: Smith, 26, who played a young political re searcher in the show, will become the biggest star of all after* ***winning*** *the role of the 11th Doctor.*

*s2: The guy is relatively unknown and the skeptics wondered if the right person was* ***chosen****.*

*האם* ***winning****ו-* ***chosen****מתייחסים לאותו האירוע.*

Complex Event Identification

*לפעמים חשוב לזהות מאורע (event) ברמה גבוהה יותר מרמת המשפט הבודד, כלומר מאורע שמתואר במספר משפטים. ניתן להגיע לאבסטרקציה של האירועים עד שמגיעים לתיאור המאורע ברמת* ***המסמך****. הרעיון הוא לסווג בצורה היררכית את המשפטים במסמך על סמך מאורעות בסיסיים יותר. הסיווג יכול שלא להתבסס על מיקום המשפט בטקסט. ניתן להתייחס לבעיה זו כבעיית clustering, כאשר כל cluster מייצג מאורע ברמה גבוהה יותר בהיררכיה. אחת הגישות לביצוע clustering כזה הוא על ידי* ***paired representation learning****. בגישה זו, אנו יוצרים ייצוג לכל משפט, כך שמשפטים השייכים לאותו complex event יהיו קרובים אחד לשני, ומשפטים שאינם שייכים לאותו complex event יהיו רחוקים. בהינתן dataset מתויג של complex events, נדגום בכל שלב זוגות של משפטים, ונבצע אופטימיזציה למודל כך שהמרחקים האוקלידיים בין הייצוגים של המשפטים שהמודל מוציא יהיו קרובים עבור זוגות מאותו complex event.*

Question Answering

*מטלה נוספת בהבנת שפה טבעית היא question answering, או השבה על שאלות. נתמקד בשלושה סוגי QA. הסוג הראשון נקרא extractive QA, בו בהינתן קטע טקסט ושאלה המודל יפיק מהטקסט את התשובה לשאלה. הסוג השני נקרא extractive QA + IDK כאשר IDK הוא ראשי תיבות של I don’t know. בסוג הזה של QA המודל יכול בנוסף להחזיר תשובה "IDK" כאשר התשובה לשאלה לא מופיעה בטקסט. הסוג השלישי הוא שאלות כן ולא (Yes/No QA).*

Extractive QA

*מטלה זו, אשר נקרא גם reading comprehension, מטרתה למצוא את ה-span של התשובה לשאלה בתוך הטקסט. למשל, בהינתן השאלה:*

What was William Johnson's Iroquois name?

*והטקסט:*

The Iroquois sent runners to the manor of William Johnson in upstate New York. The British Superintendent for Indian Affairs in the New York region and beyond, Johnson was known to the Iroquois as **Warraghiggey**, meaning "He who does great things."

*המטרה היא למצוא את המילים שבהן מופיעות התשובה – במקרה זה Warraghiggey. ההנחה ב-Extractive QA היא* ***שהתשובה מופיעה בטקסט****. הדוגמה שהבאנו לקוחה מתוך SQuAD dataset אשר מכיל 100,000 זוגות של שאלות ותשובות. הקטעים עליהם כתובות השאלות לקוחים מוויקיפדיה. ה-dataset נחשב עשיר ומגוון, וקיימות בו דוגמאות רבות לא טריוויאליות, למשל הצורך להכיר מילים נרדפות (בכ-33% מהדגימות), ידע מהעולם (9%), הבנת תחביר (64%), הסקת מסקנות ממספר משפטים שונים (14%) ועוד.*

*ניתן לפתור את הבעיה באמצעות מודל BERT. המודל, לאחר שעבר pre-training על MLM ועל NSP, יעבור fine-tuning, כאשר ה-input של המודל יהיה השאלה והקטע, כאשר מפריד ביניהם token של [SEP] (אותו token שנעשה בו שימוש ב-NSP), וה-output של המודל יהיה פרדיציה של start ו-end ל-span של התשובה. בנוסף, נוסיף לכל אחד מהtokens שבשאלה embedding שנסמן A, ובקטע B, על מנת לסמן למודל איזה מה-tokens שייכים לשאלה ואיזה שייכים לקטע.*

*את הקלסיפיקציה נבצע בצורה הבאה. נגדיר שני וקטורים (נלמדים) , ונשתמש בהם כדי לחזות את ההתחלה והסוף של ה-span. את ההתחלה נחזה באמצעות softmax על כל הembeddings של המילים שבקטע:*

*ואת הסוף נחזה באותה הצורה:*

*ה-loss במהלך האימון הוא:*

*כאשר הם המיקומים המתויגים של ההתחלה והסוף של התשובה. לטובת הפרדיקציה, נחשב ניקוד עבור כל זוג של התחלה וסוף:*

*עבור , וניקח את הזוג עם הניקוד הגבוה ביותר.*

Extractive QA with IDK

*מטלה זו דומה למטלה הקודמת, אלא שכאן השאלה יכולה להיות ללא תשובה בקטע. במקרה זה על המודל להחזיר IDN (I don’t know) על שאלה זו. SQuAD 2.0 הוא הרחבה של SQuAD 1.1, וכולל בתוכו גם 50,000 שאלות שהתשובה להן היא IDK. השאלות נכתבו על בסיס שאלות שהתשובה קיימת בטקסט אשר שונו מעט, כך שהתשובה IDK אינה טריוויאלית. שאלות אלו הן קשורות לקטע, וקיים בו "plausible answer", כלומר תשובה שהייתה יכולה להיות נכונה אם השאלה הייתה מעט שונה.*

*הגישה לפתרון בעיה זו דומה מאוד לפתרון בעיית ה-Extractive QA ללא IDK, עם שינוי אחד – הוספת חיזוי האם התשובה קיימת בקטע על בסיס token ה-[CLS] (ה-token הראשון ב-BERT). בזמן הפרדיקציה, מחושבים הביטויים:  
כאשר הביטוי הראשון שייך לחיזוי האם קיימת תשובה בקטע והביטוי השני שייך לחיזוי מיקום התשובה בקטע (במידה וקיימת). על מנת לחזות IDK מחושב:  
כאשר ניתן לבצע אופטימיזציה על על מנת למקסם את ה-recall/precision על פי דרישת המערכת.*

*גישה אחרת לפתרון היא לא לאמן על שאלות ללא תשובה, ובזמן inference לחזות IDK במידה וה-score המקסימלי קטן מסף כלשהו. לחילופין, ניתן לאמן את המודל כך שיחזה בתחילה האם קיימת תשובה בקטע (סף סטטי במקום סף דינמי כפי שהוצע בפתרון הראשון), או לאמן מודל נוסף אשר יחזה האם קיימת תשובה בקטע.*

*על מנת לבצע אבלואציה ניתן להשוות את ה-span שחזה המודל ל-span המתויג, בדומה למה שנעשה בבעיות הקודמות שהצגנו המבוססות span. SQuAD מכיל מספר אופציות ל-span עבור כל דגימה, ולכן ניתן להחשיב את התיוג כנכון אם הוא זהה לאחד מהאופציות. לאחר שנבדק האם החיזוי נכון או לא, ניתן להשתמש במדדים הסטנדרטיים: recall, precision ו-F1. בנוסף, יהיה מדד נוסף לחיזוי ה-IDK.*

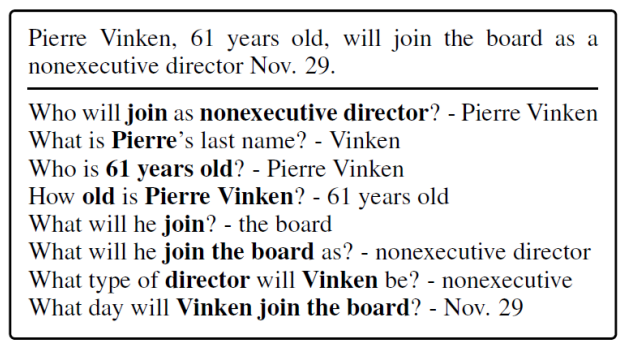
Yes/No QA

*הסוג האחרון של ה-QA שנסקור הוא שאלות "כן ולא". מטלה זו דומה למטלות הקודמות, נתון קטע ושאלה על הקטע. ההבדל הוא שהתשובה יכולה להיות רק "כן" או "לא". קיים dataset לסוג כזה של שאלות בשם BoolQ, המכיל 9,000 שאלות ותשובות.*

*ניתן למדל בעיה זו כבעיית pair-wise sentence classification, כלומר חיזוי על זוג משפטים (כמו NSP). בהינתן שני המשפטים: השאלה והקטע, מתבצעת קלסיפיקציה בינארית. גם בעיה זו ניתן לפתור באמצעות BERT בדומה ל-sequence classification.*

QAMR

*Question-Answer Meaning Representation (Michael et al., 2018), או בקיצור QAMR, הוא מאמר אשר קושר קשר מעניין בין בעיות QA לבעיות predicate-argument (בעיות מהסוג של SRL שהוזכרו לעיל). מטרת המאמר היא לענות על השאלה האם ניתן להשתמש ב-QA על מנת לבצע ניתוח סמנטי של טקסט. בעבודה זו, אספו dataset של 5,000 משפטים כאשר בכל משפט נשאלו שאלות בסיסיות על המשפט שמהם ניתן להסיק תיוגים של predicate-argument. להלן דוגמה הלקוחה מתוך המאמר:*

**

*הייתרון באיסוף data כזה הוא שלא נדרש מומחה שיתייג עבור כל משפט מהו הנשוא שלו ומהם הארגומנטים המתאימים לו (כפי שנעשה ב-SRL), אלא אלה שאלות שכל אדם ממוצע יכול לנסח ולענות עליהם. בנוסף, שאלות אלה מתארות את כל המידע הדרוש לביצוע המטלה של חיזוי predicate-argument. לאחר שמאומן מודל על ה-data הזה, ניתן לבצע fine-tuning על מנת לתייג SRL או לענות על Extractive QA.*

Textual Entailment (גרירה טקסטואלית)

*גרירה טקסטואלית היא היכולת להסיק את היחס הלוגי בין שני משפטים, או ליתר דיוק, האם משפט נובע ממשפט אחר. בדרך כלל הדרישה היא להסיק האם משפט ב'* ***נובע*** *ממשפט א',* ***סותר*** *אותו או* ***ניטראלי*** *(לא נובע אך לא סותר). בעיה זו נקראת Recognizing Textual Entailment (RTE), Textual Entailment (TE) או Natural Language Inference (NLI).*

*כל דגימה מורכבת מהנחה (premise) והיפותזה (hypothesis) , והפרדיקציה יכולה להיות אחד משלושת הקטגוריות: ENTAILMENT (נובע), CONTRADICTION (סותר) או NEUTRAL (ניטרלי). להלן דוגמאות:*

*Premise: Jane, who is a native of Los Angeles, married a lawyer from New-York.  
Hypothesis: Jane was born in Los Angeles.  
Label: ENTAILMENT*

*Premise: Jane, who is a native of Los Angeles, married a lawyer from New-York City.   
Hypothesis: Jane married in New-York City.   
Label: NEUTRAL*

*Premise: Jane, who is a native of Los Angeles, married a lawyer from New-York City.   
Hypothesis: Jane was born in France.   
Label: CONTRADICTION*

*Stanford Natural Language Inference Dataset (SNLI) הוא dataset המכיל כ-570,000 דגימות כאלה, אשר תויגו באמצעות crowdsourcing. ברוב המקרים ישנו תיוג מוסכם, אך קיימות דוגמאות בהן המתייגים לא הסכימו על התיוג. במקרים כאלה התיוג בפועל נבחר על פי הרוב. בנוסף, Multi-NLI או MNLI (Williams et al., 2018) הוא dataset מפורסם נוסף לגרירה טקסטואלית המכיל 300,000 דוגמאות בסט האימון. MNLI מכיל משפטים מ-10 מקורות שונים, ולכן נחשב למגוון. אחת ממטרותיו היא להתמקד בהסקה שקשורה לזמן ול-modality (למשל should).*

*עם זאת, קיימים מאמרים שמראים שבתהליך התיוג קיימים תוצרי לוואי ב-data כך שלעיתים ניתן להסיק את התיוג רק על פי ההיפותזה, ללא שימוש בהנחה כלל. דבר זה נובע משיטת הבניה של ה-data. בתהליך הבניה, למתייגים ניתן משפט הנחה, אשר לקוח למשל מתוך caption של תמונה (תיאור טקסטואלי של תמונה), והם צריכים לנסח שלושה משפטים, אחד שנובע מההנחה, אחד הסותר אותה ואחד ניטראלי. ניתן להראות (Gururangan et al.) כי משפטי נביעה (entailment) לעיתים רבות מכילות משפטים ללא מגדר, למשל אם בהנחה מסופר על "שתי נשים וגבר..." המתייג יכתוב במשפט הנביעה "שלושה אנשים...". לעומת זאת, משפטי סתירה (contradiction) מכילים הרבה* ***מילות שלילה****. באחת העבודות הראו כי ניתן לאמן מסווג שמגיע ל-accuracy של 67% על SNLI ו-53% על MNLI (53% על 3 classes) כאשר הוא מאומן אך ורק על ההיפותזות.*

*גם מטלה זו ניתן למדל כ-pair-wise sequence classification. ניתן להשתמש ב-BERT בדומה לדרך שתוארה עבור Yes/No QA, עם ההבדל שהתיוג הוא על שלושה classes ולא על שנים.*

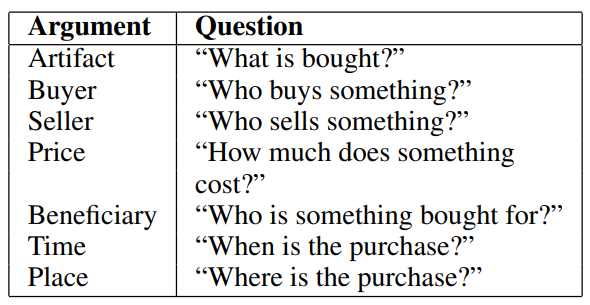
Prompting for NLU

*בפרק זה נדון בשלושה נושאים. הראשון הוא למידה באמצעות מטלות קרובות (proxy task based learning). נראה כי לעיתים ניתן להשתמש במודל שאומן על מטלה אחת, למשל מודל שאומן לענות על שאלות (QA), על מנת לפתור מטלה אחרת, למשל sentiment analysis. נראה שלפעמים ניתן להשתמש במודל ללא אימון בכלל (zero shot). הנושא השני הוא template based learning: כיצד ניתן להשתמש במודל שפה (למשל BERT)* ***שלא*** *עבר fine-tuning, על מנת לפתור משימות שונות. נראה כי ניתן להמיר מטלות שונות למטלה של השלמת מילים (MLM) ובכך לנצל את המודל ה-pretrained. והשלישי למידה באמצעות הוראות או דוגמאות (Instruction based learning).*

Proxy-Task Based Learning

*קיים אתגר בפרדיגמה הנפוצה היום של pretraining ו-fine tuning, שלמודל קשה להכליל לאחר ה-fine tuning ל-data מהתפלגות מעט שונה מזו שהמודל ראה באימון. למשל, מודל שהגיע לביצועים טובים על SQuAD לא בהכרח יביא ביצועים טובים כאשר יקבל דוגמאות על domains שהוא לא ראה (Sulem et al., 2021). דוגמה נוספת, ב-SRL המודל בדרך כלל לומד ספציפית את הפורמליזם עליו הוא אומן ולא יודע להכליל למטלות דומות.*

*בעבודה שנעשתה על ידי Lyu et al. הציעו להתמודד עם הנושא על ידי שימוש במודל מאומן למטלה אחרת שיכולה לעזור למטלה שאותה רוצים ללמוד. למשל, שימוש במודל שאומן על Textual Entailment או על Question Answering על מנת לבצע Event Extraction ב-zero shot. למשל, משתמשים במודל QA אשר יענה על שאלה מוגדרת מראש על מנת לסווג את הארגומנטים. דוגמאות לשאלות כאלה לסיווגים שונים לארגומנט עבור event של TRANSFER-OWNERSHIP ניתן למצוא בטבלה הבאה (מתוך המאמר):*

**

*מאמר אחר של Sainz et al. (2022) מראה שימוש במספר datasets של TE על מנת לאמן מודל שניתן להשתמש בו ל-Relation Extraction.*

*כפי שראינו לעיל (בפרק על SRL), במטלת Event Extraction אנו מעוניינים לזהות את ה-trigger ואת הארגומנטים של ה-trigger ולסווג אותם לקטגוריות. למשל, בקטגוריה של TRANSFER-OWNERSHIPנסווג את הקונה, המוכר, זמן הקניה, ועוד. הגישות המסורתיות, אלה שהצגנו לעיל, ניגשות לבעיה כבעיית supervised learning. גישה זו היא הפשוטה והבסיסית ביותר, וקיימים datasets רבים שתויגו לטובת המטלה, אך יש לה מספר חסרונות. ראשית התיוג יקר, שכן הוא לוקח זמן ודורש מומחי תוכן לביצוע התיוג. בנוסף, datasets שונים מתויגים תחת קונבנציות שונות (שאת חלקן ראינו לעיל), מה שגורם לכך שכל dataset יהיה מעט שונה, ויהיה קשה להכליל מאחד לשני. בנוסף, בדרך כלל קיימים תחומים סמנטיים (domains) ספציפיים לכל dataset וקשה למודל שאומן על dataset אחד להכליל לתחומים שאותם הוא לא מכיר. בעיה נוספת היא שאם נרצה להוסיף event חדש נצטרך לאסוף data על ה-event ועל המבנה שלו ולאמן שוב מודל על כל ה-data כולל ה-event החדש.*

*בגלל כל הבעיות הללו, הוצעה גישה לביצוע event extraction באמצעות מטלות אחרות, למשל QA, ללא צורך באימון מודל. הרעיון הוא שניתן להגדיר עבור כל event אוסף שאלות, אשר מהתשובות להן ניתן לבצע event extraction בצורה ישירה. ניתן באמצעות אותו מודל QA להכליל ל-datasets שונים של event extraction ולתחומים שונים שלא קיימים ב-dataset זה או אחר.*

*לצורה הדגמה, נשתמש בדוגמה שהובאה לעיל בפרק על SRL:*

*China purchased two nuclear submarines from Russia last month.*

*בשלב הראשון נזהה את ה-trigger (למשל באמצעות אלגוריתם rule based פשוט) – purchased.*

*לאחר מכן נשתמש במודל QA לסיווג ה-trigger. למשל, במודל Yes/No QA:*

* *Q: “Did someone go on vacation?”*
* *A: “No” (*⇒ *GOING-ON-VACATION X)*
* *Q: “Did someone have a birthday party?”*
* *A: “No” (*⇒ *BIRTHDAY-PARTY X)*
* *Q: “Did someone transfer ownership?”*
* *A: “Yes” (*⇒ *TRANSFER-OWNERSHIP V)*

*באותה מידה, ניתן להשתמש במודל של Textual Entailment במקום במודל Yes\No QA, על ידי הצגה של המשפט עצמו כהנחה (premise) ואת המשפט “this text is about {event type}” כהיפותזה (hypothesis). בצורה כזאת ניתן להסיק את סיווג ה-trigger. לאחר מכן, ניתן לשאול את השאלות המתאימות לסוג ה-trigger באמצעות מודל Extractive QA. למשל:*

* *Q: “What was purchased?”*
* *A: “Two nuclear submarines”. (⇒Artifact-Arg)*
* *Q: “Who purchased two nuclear submarines?”*
* *A: “China”. (⇒Buyer-Arg)*
* *Q: “Who did China purchase two nuclear submarines from?”*
* *A: “: Russia”. (⇒Seller-Arg)*

*קיימות מספר עבודות אשר בדקו גישות אלה, למשל Lyu et al. ו-Sainz et al.. גישות אלה עובדות משמעותית טוב יותר מגישות אחרות שאינן משתמשות ב-supervision, אבל קיים פער משמעותי בין הגישות הללו לגישות supervised learning. קיימים מספר מקורות לשגיאה, הן בחילוץ ה-event והן בחילוץ הארגומנטים. למשל, בחילוץ ה-event לפעמים נדרש קונטקסט ארוך יותר להבנה, דוגמה לכך ניתן לראות במשפט הבא:*

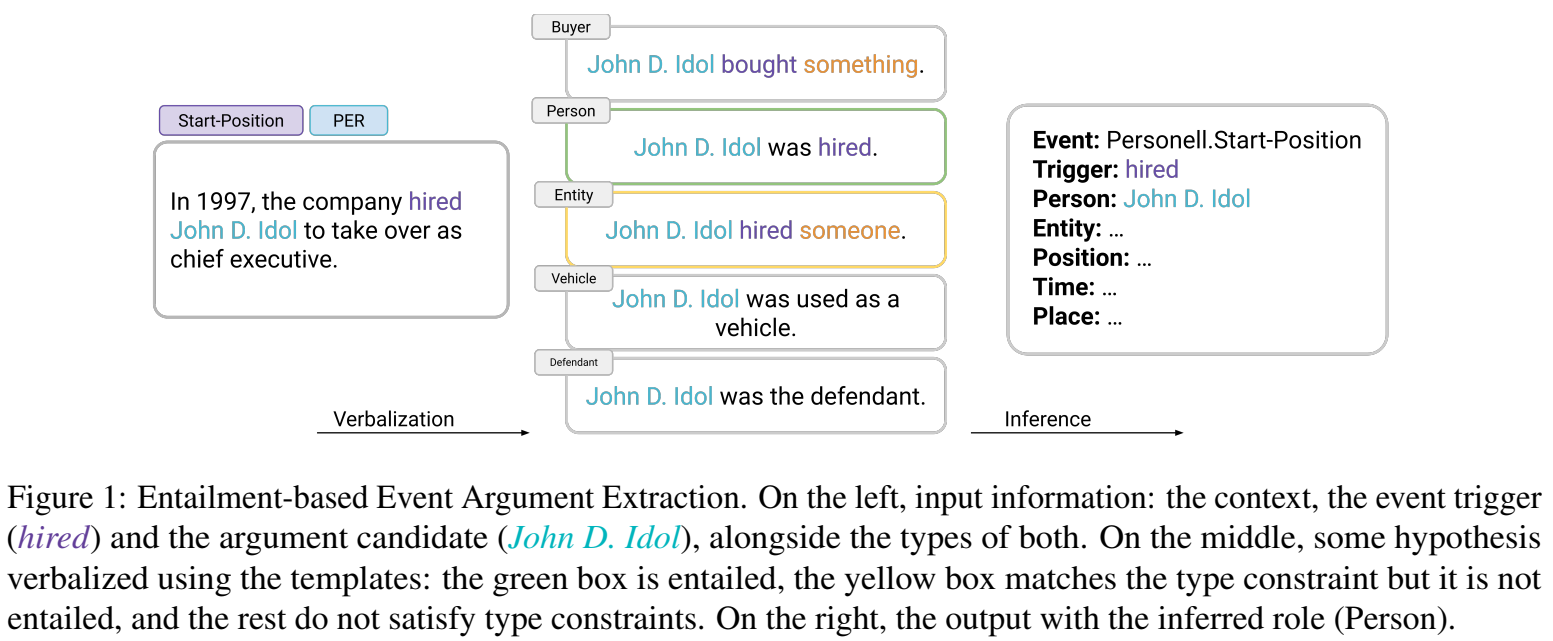
*“Turkey sent 1,000 troops ... and said* ***it would send more.****”*

*אם המודל רואה רק את המילים המודגשות על מנת לסווג את המאורע (כאשר ה-trigger הוא send), המודל עלול לסווג את המאורע כ-TRANSFER-MONEY, כאשר הסיווג הנכון הוא TRANSPORT, אך ניתן להגיע למסקנה זו רק אם רואים את תחילת המשפט. מקור נוסף לשגיאות הוא מקרים בהם ישנה יותר ממשמעות אחת למשפט. למשל:*

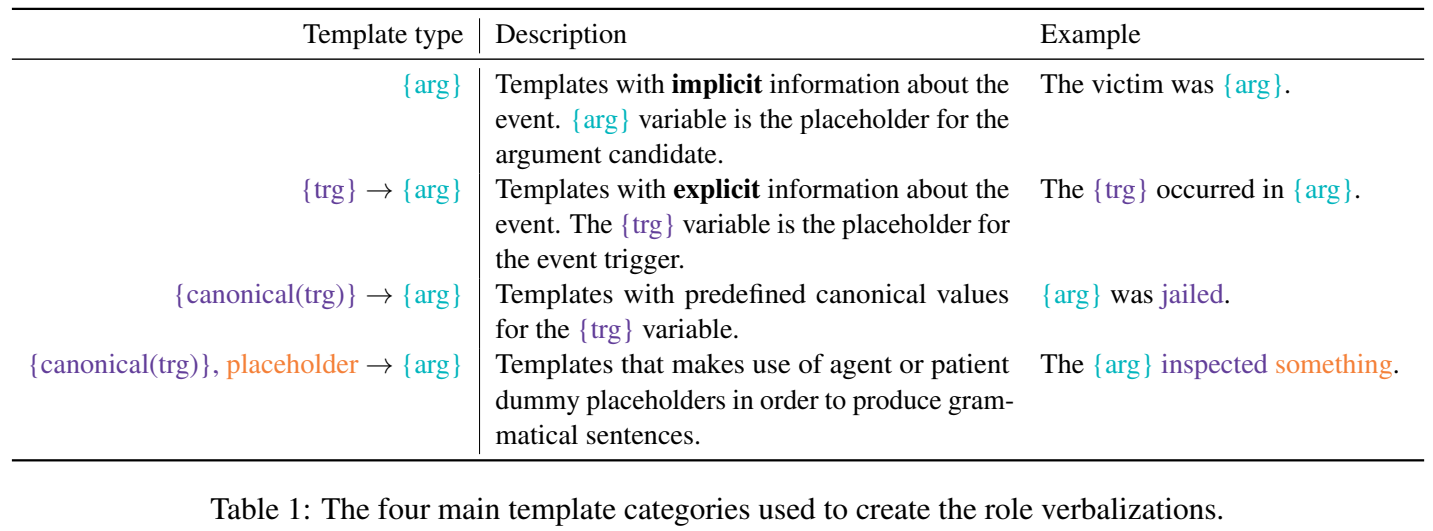
*“A unit meets in confidential sessions to review terrorist activities* ***in Europe****.”*

*הביטוי “in Europe” יכול להתייחס במקרה זה לפגישה (meets in Europe) או לפעולות הטרור (terrorist activities in Europe). כאשר שואלים את המודל "איפה הפגישה קרתה?" המודל עלול לענות "in Europe" למרות שזו לא התשובה הנכונה. בנוסף השאלות עלולות להיות מנוסחות לא נכון, ויתכנו בעיות נוספות.*

*עבודה נוספת שנעשתה על ידי Sainz et al. (2022) מראה שימוש ב-TE לצורך חילוץ ארגומנטים (argument extraction). בהינתן מודל TE מאומן, ואלגוריתם שיודע לחלץ את ה-trigger ומועמד לארגומנט, ניתן להציב את ה-trigger והארגומט בתוך טמפלייטים שונים, ולבדוק באמצעות מודל ה-TE האם המשפט שנוצר נובע מהמשפט המקורי.*

**

*להלן מספר templates שנעשה בהם שימוש במאמר:*

**

*עוד דבר מעניין שהראו במאמר הוא שכאשר מודל ה-TE אומן על מספר datasets שונים הוא מכליל טוב יותר ומניב ביצועים טובים יותר במשימת חילוץ הארגומנטים. להלן ההבדלים בביצועים על datasets שונים, כאשר ההשוואה נעשתה בין מודל TE שאומן על MNLI בלבד מול מודל שאומן על MLNI, FEVER, SNLI ו-ANLI. ההבדל בין העמודות הוא האם הבדיקה נעשתה ב-zero shot (0%), ב-few shot (5%) או באימון supervised מלא (100%).*

*A screenshot of a table

Description automatically generated*

*ANLI הוא dataset שנועד להכיל דוגמאות קשות ל-NLI. במהלך בניית ה-dataset המתייגים נדרשו למצוא דוגמאות שבהם המודל הטוב ביותר שהיה קיים יטעה. לאחר שנאספו דוגמאות כאלה, אומן מודל מחדש ושוב ניתן למתייגים למצוא דוגמאות בהם המודל החדש טועה. תהליך זה התבצע מספר פעמים כך שנאספו דוגמאות קשות. FEVER הוא dataset לחילוץ ובדיקה של עובדות (Fact Extraction and VERification). שלושת ה-classes הם: SUPPORTED, REFUTED ו-NOT ENOUGH INFO.*

*עד כה תיארנו כיצד ניתן להשתמש במטלה אחת (TE) על מנת לפתור מטלה אחרת (event extraction). אך ניתן להשתמש באותו הרעיון גם במטלות אחרות. למשל, ניתן לפתור את מטלת relation extraction (זיהוי הקשרים שבין הישויות במשפט, כפי שדנו לעיל) באמצעות מודל QA (Li et al., 2019) או מודל TE (Sainz et al., 2021). דוגמה נוספת היא ביצוע סיווג של נושא (על איזה נושא מדובר במשפט) או רגשות (האם המשפט מבטא רגש חיובי או שלילי) באמצעות TE (Yin et al., 2019) או QA (Puri and Catanzaro, 2019). בנוסף, ניתן לבצע coreference resolution (עיין לעיל) באמצעות QA (Wu et al., 2020).*

Template Based Learning

*ניתן להמיר מטלות שונות של NLU לצורה שניתן לפתור באמצעות* ***מודל שפה שעבר pre-training בלבד****. ניתן דוגמה פשוטה. נניח מודל BERT שאומן על MLM, ניתן להשתמש בו לביצוע sentiment analysis. למשל, עבור חיזוי על המשפט:*

*Best pizza ever!!!*

*ניתן להמיר את המשפט למשפט:*

*Best pizza ever!!! It was [MASK]*

*להזין אותו ל-BERT ולראות את ההשלמה שלו ל-[MASK]. במידה וההשלמה היא מילה חיובית (למשל great) ניתן להסיק שהרגש של המשפט חיובי, ולהיפך. Schick and Schütze, (2021) הראו שניתן בצורה זו לבצע* ***TE****. למשל עבור הדגימה הבאה:*

Premise: Mia likes pie  
Hypothesis: Mia hates pie  
  
Formulation: Mia likes pie? [mask], Mia hates pie

*המשפט שמופיע ב-formulation מוזן ל-BERT, ונבדק ה-softmax של ה-tokens: Yes, No, Maybe. על פי ההסתברויות הללו ניתן להסיק האם המשפט נובע מההנחה (Yes), סותר (No) או ניטראלי (Maybe).*

*Chen et al. (2020) הראו שניתן בשיטה זו למדל את מטלת* ***argument extraction****. למשל עבור המשפט הבא:*

Americans sought to bring calm to Mosul, where U.S. troops killed 17 people in clashes earlier in the week.

*ניתן להוסיף לסוף המשפט את הטקסט:*

[Someone] killed [someone] with [something] in [some place] at [some time]

*ולהזין את שני המשפטים אחד אחרי השני ל-BERT. לאחר מכן, ניתן לראות את ההשלמות של BERT. ההשלמה יכולה להיות מהצורה:*

U.S. troops killed 17 people with something in Mosul at earlier in the week.

*וכך ניתן לשייך כל ביטוי שהופיע במשפט המקורי לסוג ארגומנט על פי המקום ב-template בו BERT השלים.*

***בדיקה של קשרים או עובדות*** *(*Probing for relations/facts), ניתן לבצע באותה צורה (Petroni et al., 2019). למשל ניתן לנסח את המשפט:

Dante was born in \_\_\_\_.

*ולבדוק את ההשלמה של BERT.*

*וכן מטלות של* ***commonsense*** *(*Trinh and Le, 2019), למשל במשפט הבא:

The trophy doesn’t fit in the suitcase because it is too big.

מטלת commonsense יכולה להיות, האם המילה it מתייחסת ל-trophy או ל-suitcase. מהתחביר של המשפט לא ניתן להסיק, ונדרש "שכל ישר" (common sense) לפתרון הבעיה, שכן לא יכול להיות שלא ניתן להכניס משהו למזוודה מכיוון שהמזוודה גדולה מידי. ניתן לראות מהי הפרדיציה של המודל עבור ה-token it, האם ה-softmax גבוה יותר במילה trophy או suitcase.

בדומה לכך, ניתן לבצע מטלת **reasoning** (Talmor et al., 2020), למשל:

The size of an airplane is [MASK] than the size of a house. A. larger B. smaller

*המודל יחזה שה-token הממוסך מתאים יותר ל-larger מאשר smaller.*

*להשלים:* Template-based learning – In-depth example

Instruction Based Learning

*רוב הפעמים שדיברנו עד כה על מודל שפה התייחסנו בעיקר למודלים מסוג encoder, כמו BERT, אשר מקבלים משפט והמודל חוזה ייצוג לכל token במשפט, וכן לכל המשפט (דרך token ה-[CLS]). כעת נראה כיצד ניתן להשתמש במודלי שפה* ***גנרטיביים****, אשר אומנו לחזות את ה-token הבא בהינתן תחילת המשפט. מודלים כאלה, כאשר הם מבוססים טרנספורמר, משתמשים בdecoder ולא ב-encoder כמו BERT. GPT היא משפחה של טרנספורמרים כאלה (decoders) אשר אומנו על מספר הולך וגדל של tokens. בעוד ש-GPT המקורי (Radford et al., 2018) אומן על 800,000 tokens, ו-GPT-2 (*Radford et al., 2019*) אומן על 10,000,000 טוקנים (8 מיליון מסמכים ודפי אינטרנט), GPT-3 (*Brown et al., 2020*) אומן על 500 מיליארד tokens. GPT-2 היה יכול לבצע משימות מוגדרות (למשל – כתוב סיפור על...) וכל לסכם טקסטים. ב-GPT-3 גילו שניתן לבצע משימות מורכבות באמצעות הצגה של מספר דוגמאות למודל.*

*ניתן להשתמש ב-GPT-3 לפתרון בעיות ב-zero shot, one shot ו-few shot. ב-zero shot ניתן למודל הוראות (instructions) בתוך template וניתן לו להשלים את התשובה. למשל ניתן למודל את הטקסט הבא (prompt):*

Translate English to French:

Cheese =>

*ונראה מהו ההמשך שהמודל מג'נרט. על מנת לשפר את המודל, ניתן לתת למודל דוגמה אחת (one shot) או מספר דוגמאות (few shot) לפתרון, למשל:*

Translate English to French:

sea otter => loutre de mer

peppermint => menthe poivrée

plush giraffe => giraphe peluche

Cheese =>

|  |  |  |
| --- | --- | --- |
| ***Yes/No QA [F1]*** | ***Extractive QA [F1]*** | ***Setting*** |
| *91.0* | *93.0* | *Fine tuned SOTA* |
| *60.5* | *59.5* | *GPT-3 zero shot* |
| *76.7* | *65.4* | *GPT-3 one shot* |
| *76.4* | *69.8* | *GPT-3 few shot* |

*מספר מאמרים הראו כי אף על פי שהביצועים של המודל לא טובים כמו מודל fine-tuned למשימה, הביצועים לא רעים ב-few shot ואפילו ב-one shot. להלן השוואת ביצועים על Extractive QA (SQuAD 2.0) ועל Yes/No QA (BoolQ):*

*לשם השוואה, ביצועי BERT-large* ***שעבר fine tuning*** *על BoolQ מגיע ל-77.4%.*

*אחת הבעיות בשימוש במודל שאומן אך ורק על חיזוי המילה הבאה היא שהחיזוי אינו יציב. כלומר, אין מה שיכווין את המודל (מלבד הנדסת ה-prompt) לענות דווקא את התשובה שאנו מעוניינים. לדוגמה, עבור ה-prompt הבא:*

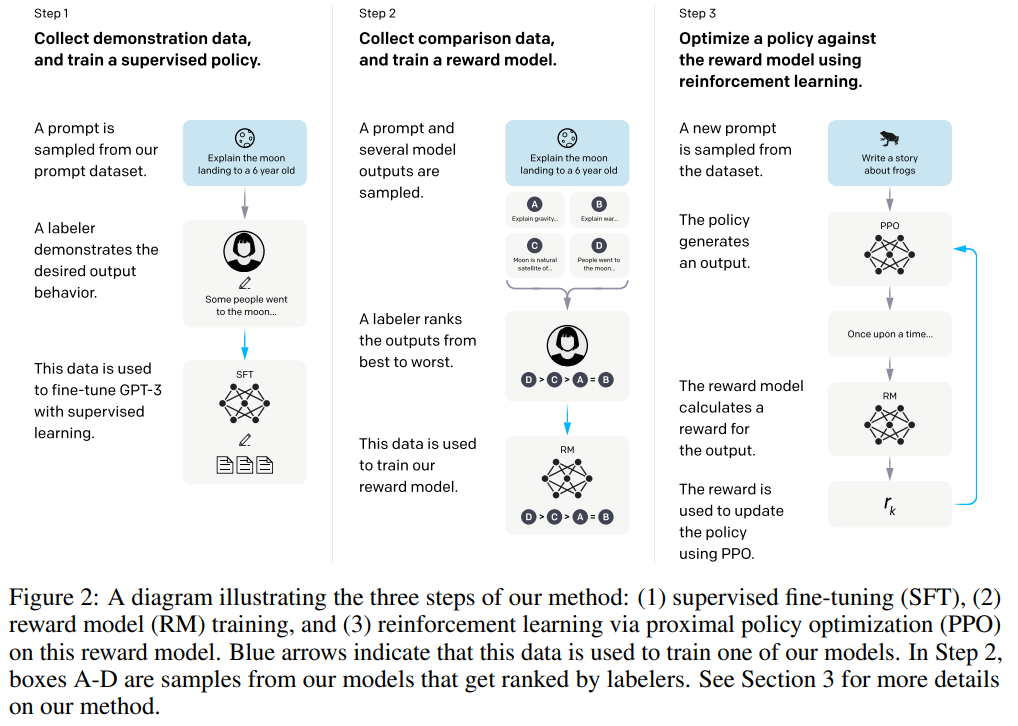
*Write a story about a frog traveling in space.*

*התשובה של GPT עלולה להיות:*

*Write a story about a horse traveling in space.*

*Write a story about a dog traveling in space.*

*דבר זה כנראה נובע מכך שהמודל ראה מסמכים של שאלות, ולכן כאשר הוא רואה שאלה הוא יודע שסטטיסטית יש סיכוי גבוה שתהיינה שאלות עוקבות. פתרון שהוצע על ידי* Ouyang et al. (2022) *היה לבצע fine tuning למודל כך שיעקוב אחרי הוראות. סכמת האימון מופיעה באיור הבא:*

**

*בשלב הראשון, המודל עובר fine tuning על data מתויג של שאלות (instructions) ותשובות, כאשר התשובות נכתבו על ידי מתייג אנושי. בשלב השני, עבור כל שאלה נדגמות מספר תשובות ממודלים שונים, ומתייג אנושי מדרג את התשובות. על בסיס תיוג זה מאומן מודל שחוזה עד כמה התשובה טובה. בשלב האחרון, מבצעים fine tuning נוסף ל-GPT באמצעות* ***reinforcement learning****, וספציפית באמצעות אלגוריתם PPO, כאשר ה-reward מתקבל באמצעות המודל שאומן בשלב השני, כך שהמודל יפיק את התשובות הטובות ביותר. במאמר הראו שבאמצעות אימון כזה המודל נוטה לענות בהסתברות גבוהה הרבה יותר את מה שהיינו מצפים שיענה.*

QA as a Meaning Representation

*מספר עבודות נעשו על מנת להעשיר את הייצוג של ה-tokens ש-BERT לומד. אחד מהכיוונים הוא להתבסס על מידע סמנטי של משפטים וכן על משימות בסגנון של SRL על מנת ללמוד ווקטור embedding שיכיל יותר מידע וכך יגיע לביצועים טובים יותר במגוון משימות. לצורך כך ניתן להשתמש ב-dataset QAMR (אשר תיארנו לעיל). Dataset זה מכיל שאלות ותשובות על המידע הסמנטי של משפט, למשל:*

*Source Sentence: "Sheila baked a delicious cake."  
Target Sentence (Question): "Who baked the cake?"  
Answer: "Sheila"*

*Source Sentence: "The concert lasted for three hours."  
Target Sentence (Question): "How long did the concert last?"  
Answer: "three hours"*

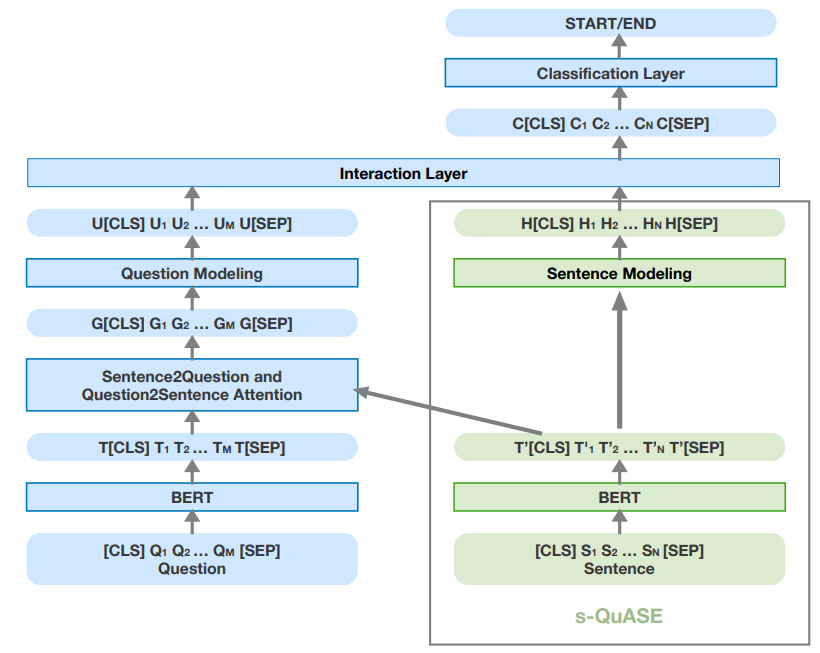
*Source Sentence: "I visited Paris during my summer vacation."  
Target Sentence (Question): "Where did you visit during your summer vacation?"  
Answer: "Paris"*

*Source Sentence: "The book was written by J.K. Rowling."  
Target Sentence (Question): "Who wrote the book?"  
Answer: "J.K. Rowling"*

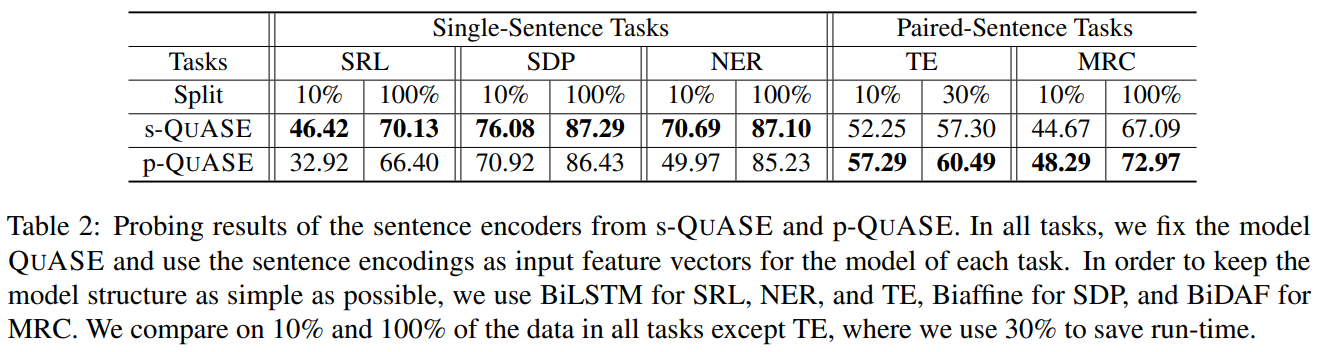
*ההנחה היא ש-dataset כזה מכיל אינפורמציה סמנטית של השפה, ולכן שימוש בו במודל שפה עשוי לשפר את הביצועים שלו. רעיון זה דומה לרעיון שבסיס SemBERT שנסקר בפרק על SRL. שם הראו ששימוש ב-SRL במודל שפה בנוסף למשפט עצמו משפר ביצועים במטלות שונות.*

*ישנן שתי אלטרנטיבות כיצד לבצע את הלמידה של QAMR: לאחר ה-pre-training הראשוני או במקביל אליו. בגישה הראשונה, שהיא גם הפשוטה יותר, מאמנים בשלב ראשון את BERT על מטלת pre-training (כגון MLM ו-NSP), לאחר מכן מבצעים fine tuning ראשוני על QAMR, כאשר מתייחסים לבעיה כבעיית Extractive QA, ולבסוף מבצעים fine tuning נוסף על המטלה הרצויה.* Supplementary Training on Intermediate Labeled data Tasks (STILTs) *היא שיטה שהוצגה במאמר של* Phang et al. (2018), אשר מראה שעל ידי fine tuning ראשונים על מטלות supervised שונות שקיים להן data ניתן לשפר מודלי שפה למטלות אחרות. במאמר זה הוצג ה-framework של ביצוע fine tuning ראשוני על מטלות אחרות. בנוסף, Clark et al., 2019 הראו שיפור על ה-dataset של BoolQ באמצעות אימון מקדים על MNLI.

האלטרנטיבה השנייה להעשרת מודל שפה באמצעות מטלות הבנת שפה הוצגה במאמר בשם QUASE: Question-Answer Driven Sentence Encoding על ידי He et al. (2020). במאמר טוענים שביצוע fine tuning פשוט על המטרה במקרים מסוימים אינו מספיק, והציעו שיטה בשם שמביאה לביצועים טובים יותר. במאמר מבחינים בין מטלות כמו SRL ו-NER אשר לא דורשות הבנת קשר בין שני טקסטים, לבין מטלות כמו QA ו-TE אשר דורשות הבנה של קשר בין שני הטקסטים (הטקסט והשאלה ב-QA וההנחה וההיפותזה ב-TE). במאמר מראים כי אימון מקדים על dataset של שאלות ותשובות כמו QAMR אכן משפר מטלות מהסוג השני, אך לא מהסוג הראשון. דבר זה כנראה קורה מכיוון שמטלת QA בעצמה היא מטלה המתבססת על למידת היחס בין שני משפטים, ולכן משפרת מטלות כאלה. לשיטת ה-fine tuning שהוצגה במאמר של STILTs נתנו את השם p-QUASE, ראשי תיבות של **paired** question-answer driven sentence encoding. החידוש במאמר זה הוא פיתוח שיטה בשם s-QUASE (s ל-single), אשר משפרת ביצועים למטלות של משפט אחד. להלן הארכיטקטורה המוצעת עבור s-QUASE:

**

*כפי שניתן לראות באיור, בזמן האימון על ה-data של ה-QA, השאלה והקטע עליו נשאלת השאלה עוברות בענפים נפרדים בארכיטקטורה. הארכיטקטורה שבתוך המלבן האפור היא זו שייעשה בה שימוש ל-target task, וכפי שניתן לראות אין בה אינטרקציה ישירה עם השאלה (ישנה כמובן אינטרקציה עקיפה בפעפוע הגרדיאנטים, אך אין התבססות על השאלה בזמן מידול הטקסט שבמשפט). על ידי הפרדה זו, ניתן מצד אחד ללמוד מ-data של שאלות ותשובות המכילות שני קטעים לכל דגימה, ומצד שני ללמוד ייצוג נפרד למשפט אחד אשר יעזור למטלות בהם הייצוג הדרוש הוא למשפט יחיד. כפי שניתן לראות בטבלה שלהלן, הביצועים של s-QUASE טובים יותר על מטלות על משפט אחד, והביצועים של p-SQUASE טובים יותר על מטלות של שני משפטים:*

**

UCCA Annotation

מודל ה-UCCA

*אחד ממטרות היסוד של הבנת שפה טבעית היא להצליח ליצור ייצוגים טובים ואינפורמטיביים לסמנטיקה של השפה. בהינתן ייצוגים טובים, ניתן לפתור הרבה מתתי הבעיות שבתחום. ניתן לחלק את סוגי הייצוגים ל-* *Distributional semantics ול-Symbolic Semantic Representations. Distributional semantics הוא שם כולל לכל השיטות אשר מנסות ללמוד* ***ייצוג וקטורי*** *באמצעות* ***התפלגות אמפירית של data.*** *הרעיון של הגישה הוא שניתן לחלץ סמנטיקה באמצעות מודלים סטטיסטיים על כמות גדולה מספיק של טקסטים, והמטרה היא ללמוד ייצוג וקטורי כך שמילים בעלות משמעות סמנטית דומה יהיו קרובות במרחב הייצוג הוקטורי. הדוגמאות הקלאסיות לגישה זו הן Word2Vec ו-BERT. לעומת זאת, גישת הייצוג הסמנטי הסימבולי מנסה למצוא את הייצוג באמצעות* ***מבנה המשפט וחוקי השפה****. גישות אלו ינסו למדל את השפה באמצעות חוקים מוגדרים מראש ולא באמצעות ניתוח סטטיסטי, והייצוג יהיה בדרך כלל באמצעות משפטים לוגיים או knowledge graph.*

*עד כה התמקדנו בעיקר ב-distributional semantics, אך ישנם יתרונות גם בייצוג סימבולי. ראשית, ייצוג כזה תומך ב****הרכבה****. כלומר, בהינתן שני משפטים והייצוג הסימבולי שלהם, ניתן בקלות למצוא את הייצוג הסימבולי של אוסף המשפטים. בנוסף, ייצוג כזה הוא* ***אינטרפרטבילי****, בשונה מייצוג וקטורי אשר קשה לנו בתור בני אדם להבין את המשמעות שלו. ייתרון נוסף בייצוג סימבולי הוא* ***ההכללה בין שפות****. במקרים רבים ניתן להגיע לאותו הייצוג עבור שפות שונות, ולכן המעבר בין שפה לשפה הוא הרבה יותר קל. לעומת זאת, בייצוג וקטורי, נדרש ללמוד את ההתפלגות עבור כל שפה בנפרד. גם אם נעדיף להשתמש במודלים מבוססי התפלגות (distributional semantics), ניתן לשפר את ביצועי המודלים על ידי אינטגרציה של מודל סימבולי.*

*אחד מהאופציות לייצוג סמנטי סימבולי הוא ייצוג סמנטי מבוסס גרף. דוגמאות לייצוגים כאלה ניתן למצוא ב-AMR (*Abstract Meaning Representation) וב-UCCA (Universal Cognitive Conceptual Annotation). בייצוג כזה, קיימים קשרים בין ישויות בכל הטקסט, בשונה למשל מ-SRL שם הקשרים קיימים רק בתוך המשפט, ואין קשרים מחוץ לגבולות המשפט. ניתן לומר כי SRL יותר קרוב לתחביר של המשפט, וייצוגים כמו AMR יכולים להיות מנותקים מהתחביר ולהתרכז בסמנטיקה ובמשמעות. למשל, ב-AMR, כל ישות תמודל כקודקוד בגרף וכל קשר בין שני ישויות ימודל כקשת. לעומת זאת, ב-SRL, התיוג הוא לפי word span ואין מידול של הקשר בין הארגומט לנשוא. לכן SRL יכול למדל רק משפט בודד, ולא יכול למדל ארגומנט מחולק (אלא רק כשהוא בא ברצף). הבדל נוסף הוא ש-SRL קשור מאוד לתחביר, שכן זהו תיוג של משפט ספציפי ושל span של הנשוא והארגומנטים במשפט. לעומת זאת, AMR ממיר את המשפט לגרף כך שהמשפט מאבד את המבנה שלו. לצורך העניין, שני משפטים יכולים להיות ממופים לאותו הגרף. AMR קרוב יותר **למשמעות** מאשר לתחביר. בנוסף, SRL נותן תיוג של תפקיד (ארגומנטים) עבור כל אחד מהנשואים, לעומת זאת AMR יוצר גרף אחד למשפט השלם אשר מאגד את המשמעות השלמה. למשל עבור המשפט המורכב:

The cat that the dog chased ran quickly.

SRL יתייג כל פועל בנפרד:

[The cat] **ran**  
[The cat] that [the dog] **chased**

לעומת זאת, AMR יבנה גרף אחוד אשר מכיל את המשמעות של המשפט השלם. הבדל אחרון הוא ש-SRL מתייג על פי tokens של המשפט עצמו, ואילו AMR מתרכז בישויות ובמשמעות שלהן בתיוג. למשל, במשפט:

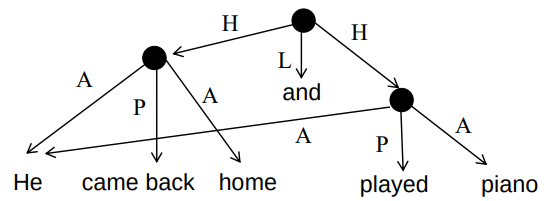
*[The children]Agent [are playing]Predicate [in the park]Location*

*התיוג של SRL הוא למילים הספציפיות המופיעות במשפט, ואילו AMR יבנה גרף עם הקונספטים: play, child, park.*

*התיוג של UCCA מבוסס בעיקרו על Cognitive Linguistics – תחום שהתפתח החל מהמחצית השנייה של המאה ה-20. בשונה מהבלשנות המסורתית (בחצי הראשון של המאה ה-20) אשר התמקדה ב****תחביר*** *ו****בחוקים*** *למידול השפה,* בלשנות קוגניטיבית (Cognitive Linguistics) באה בגישה של תפיסת השפה **על ידי האדם**. אחד מהספרים הבסיסיים בתחום הוא Cognitive Grammar: A Basic Introduction ((Langacker, 2008 אשר מניח את היסודות למידול של השפה בראי התפיסה של האדם ומדעי הקוגניציה. התיוג של UCCA, אשר מתרחק מההקשר התחבירי ומתמקד במשמעות הסמנטית של השפה, מתבסס על תפיסה זו. ספר משמעותי נוסף הוא Basic Linguistic Theory (Dixon et al., 2010, 2012) אשר חוקר טופולוגיה: סיווג של שפות שונות על פי המבנה הסמנטי שלהן.

בדומה ל-AMR, גם UCCA מנסה לבצע אבסטרקציה מעל התחביר, כלומר לייצג את המשמעות במנותק מהתחביר. המידול של השפה מתבצע ב-high level, ונותן שלד סמנטי של המשפטים והשפה. אחד מהיתרונות ב-UCCA הוא שהייצוג גם "אינווריאנטי לשפה", כלומר אותו משפט בשתי שפות שונות יקבל את אותו הייצוג, וניתן להשתמש באותן קטגוריות ואותם עקרונות לתיוג בכל שפה.

המבנה של UCCA הוא גרף כיווני ללא מעגלים (DAG), כאשר המילים מופיעות כעלים בגרף. הקודקודים בגרף נקראים units הם יכולים להיות או עלים או אוסף של אלמנטים (קודקודים אחרים) אשר מייצגות ישות יחידה. למשל בתרשים הבא:



מלבד העלים, ישנה יחידה (unit) של “He came back home”, יחידה של “He played piano”, ויחידה המכילה את כל המשפט “He came back home and played piano”. משפט כזה ממודל בצורה טבעית יותר על ידי UCCA מאשר SRL. הקשתות בגרף מייצגות את הקשרים בין האבות לבנים. בתרשים הנ"ל מופיעים ארבעה סוגי קשרים:

* H – parallel scene
* L – linker
* A – participant
* P – process

המקבילה לנשוא (predicate) ב-SRL היא ה-**Main** **relation**, אשר יכולה להיות תהליך (process, כמו בדוגמה לעיל) או מצב (stats) – למשל "She **has** a new car" או "The man **is** tall". Main relation יוצר **סצנה** (scene) – אוסף הישויות המחוברות ביחד באותו הקשר. בכל סצנה יהיו **משתתפים** (participants), למשל “he” ו-“home” במשפט שלעיל “he came back home”. נשים לב שמשתתף יכול להיות משויך למספר סצנות, למשל בדוגמה לעיל, “he” משויך גם לסצנה “he came back home” וגם ל-“he played piano”.

בנוסף לקשרים שתיארנו, “adverbial” הוא קשר נוסף אשר מתאר **קשרים משניים** (**secondary relations**), למשל במשפט “He began playing piano”, המילה began מוסיפה אינפורמציה ל-main relation (לתהליך או למצב).

כל יחידה (unit) ניתן לחלק **למרכז**, או מרכזים, ו-**elaborators** (אולי נכון לתרגם "מפרטים" או "פרטים"). למשל, את היחידה “the little boy” ניתן לחלק למרכז – “boy” ול-elaborators – “the” ו-“little”.

הייתרון בשימוש בגרף הוא שניתן לחבר תתי גרפים בצורה היררכית. למשל, סצנה יכולה להיות משתתף (participant) של סצנה אחרת:

[[Messi’s]A [kick]P]A [saved]P [the game]A

בדוגמה זו, “Messi’s kick” היא סצנה בפני עצמה, עם main relation (הבעיטה) ומשתתפים (מסי), והיא גם משתתף בהיררכיה הגדולה יותר, בה המילה “saved” היא ה-main relation. בנוסף, סצנה יכולה להיות elaborator, למשל בדוגמה הבאה:

[the]E [dog]C [[I]A [saw]P [yesterday]T ]E .

“I saw yesterday” הוא יחידה בפני עצמה, אך גם elaborator של “dog” שכן משפט זה עוזר לתאר את הכלב.

לבסוף, יכולות להיות סצנות **מקבילות**, למשל הסצנות “he came back home” ו-“he played piano” בתרשים שלעיל, כלומר שני החלקים שווים מבחינת היררכית, וניתן לחבר אותם באמצעות **linker**, “and” בדוגמה שלעיל. דוגמאות נוספות:

* John went to the store **and** Mary went to the bank.
* The kids played in the yard **while** the dog slept.
* Bob likes swimming **but** he hates running.

עוד דוגמה לך ש-UCCA ממדל את השפה בצורה אבסטרקטית יותר הוא שפועל יכול להיות בתפקיד "תואר הפועל" (adverbial). למשל, במשפט הבא:

“John began swimming”.

על פי ניתוח תחביר מסורתי, began הוא הנשוא, ולמשל SRL של המשפט הוא:

]John Arg0] ]began Predicate] [swimming Arg1]

עם כי מבחינת משמעות המשפט, המילה swimming היא המילה "החשובה". ב-UCCA, התיוג של הקשת שמקשרת את began ליחידה שלה הוא Adverbial (תואר הפועל), והמילה swimming היא ה-Process, הרכיב המרכזי במשפט. בנוסף, התיוג של **מילות יחס** (למשל in, on, to, of, for, with) הוא לפי התפקיד הסמנטי שלהם, ולא כולם מתויגות באותה הצורה. למשל:

**AfterL** graduation, he moved to NYC.

The tree is **inS** the garden.

L הוא Linker – מחבר בין שני חלקי משפט, ו-S הוא Scene evoker – רכיב ה"מעורר"/"מזמן" (evoke) סצנה שלמה.

עוד יתרון בתיוג UCCA הוא היכולת **להכליל לשפות שונות**. למשל, המשפטים הבאים:

English: “John usually comes”

Français: “John a l’habitude de venir”

למרות שבאנגלית “usually” הוא תואר הפועל, ובצרפתית “l’habitude” הוא שם עצם (תרגום מילולי: a l’habitude – יש לו הרגל, כאשר הרגל הוא שם עצם), תיוג ה-UCCA זהה – D (adverbial). מכיוון שהתיוג הוא על פי **המשמעות האנושית** ולא על פי כללי תחביר ניתן להשתמש בתיוג אחוד לכל השפות האנושיות.

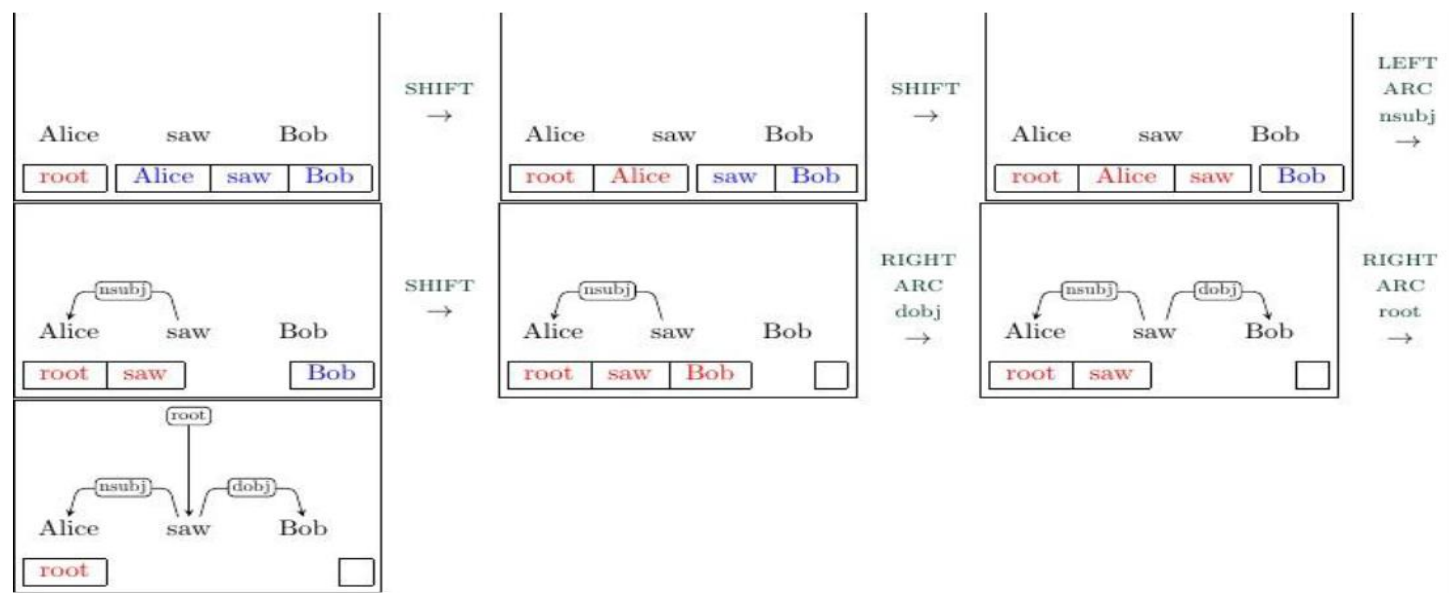
אלגוריתם תיוג UCCA

לאחר שתיארנו את מודל ה-UCCA ניתן לדון במטלת התיוג של UCCA. המטלה היא שבהינתן משפט, האלגוריתם יחזיר גרף המתאים לתיוג ה-UCCA. פתרון אפשרי הוא שימוש באלגוריתם Incremental parsing algorithm אשר מוצג במאמר A Transition-Based Directed Acyclic Graph Parser for UCCA (Hershcovich et al., 2017), אשר מבצע את הבנייה של הגרף בצורה איטרטיבית מילה-מילה. בכל שלב באלגוריתם נוצרת קשת חדשה המחברת בין שתי יחידות. נעשה שימוש באותו האלגוריתם לבניית עץ תחביר (syntax tree).

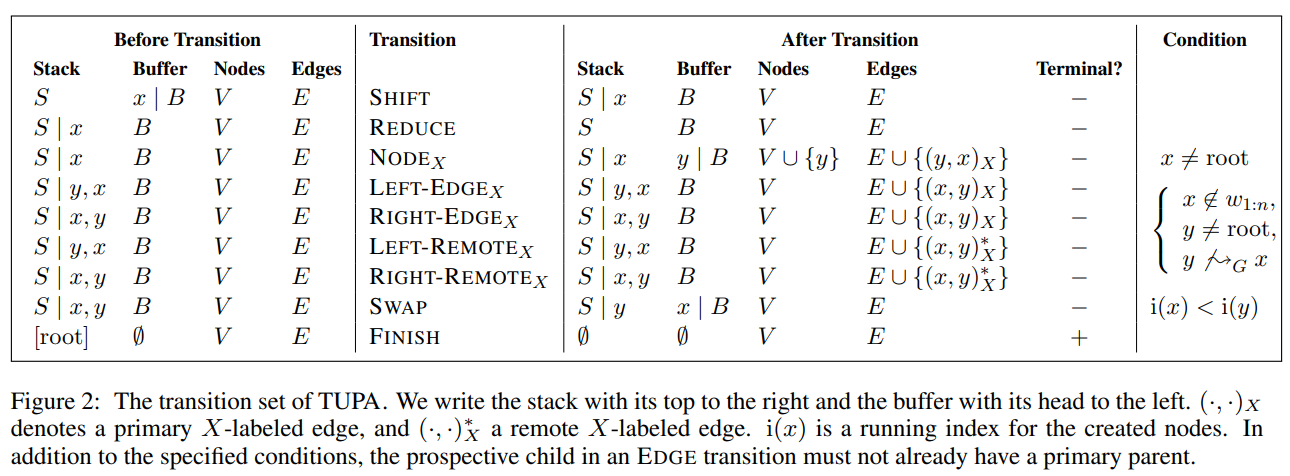
לצורך הדוגמה, נציג את האלגוריתם עבור מטלה של dependency parsing (בניית עץ תחביר). באלגוריתם זה, מאתחלים שתי רשימות – stack ו-buffer. ה-stack הוא העץ שנבנה בתהליך, ומתחיל ב-root, וה-buffer מאותחל בכל המילים במשפט. הפעולות האפשריות הן:

* SHIFT – העברה של מילה מה-buffer ל-stack, כלומר הוספה של המילה לעץ.
* LEFT/RIGHT-ARC – הוספת קשת, ימנית או שמאלית, בין שני האיברים הימניים ביותר ב-stack (בעץ), והסרה של הבן. בנוסף, האלגוריתם נדרש לבחור תיוג לקשת.

העץ נבנה מלמטה למעלה, כאשר בכל שלב מחברים בין שני מילים ומסירים את המילה שהיא הבן. להלן הדגמה של האלגוריתם על dependency parsing:

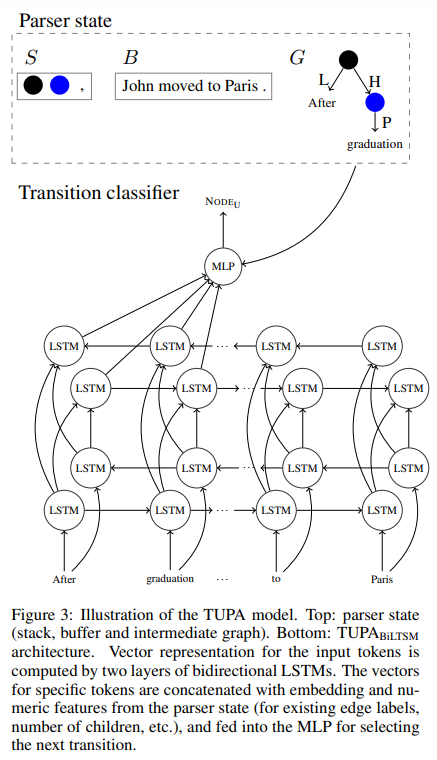


במאמר, מציגים שיטה בשם TUPA אשר מתמודדת עם תיוג UCCA בצורה דומה. האלגוריתם מתחיל עם רשימת המילים במשפט, וקודקוד המשמש כשורש, ויכול לבצע פעולות שונות עד לתוצאה הסופית. להלן תרשים מתוך המאמר אשר מתאר את אוסף הפעולות החוקיות של האלגוריתם:

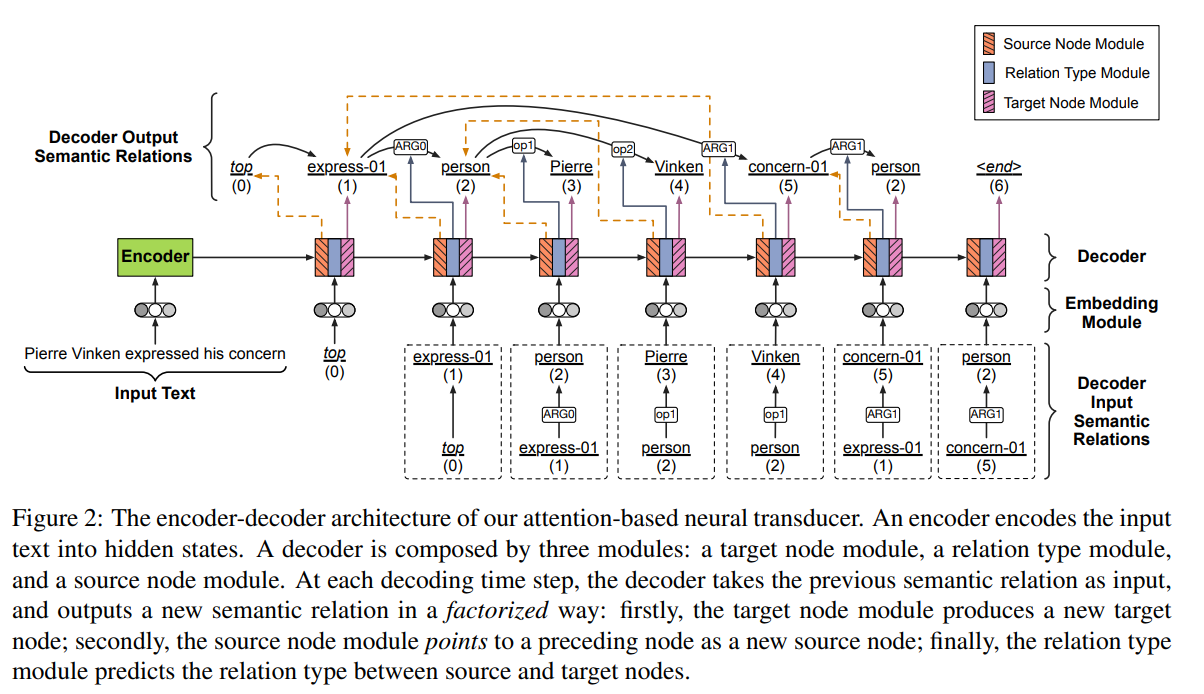


פעולת **SHIFT** מעבירה איבר מתחילת ה-buffer לסוף ה-stack. פעולת **REDUCE** מסירה את האיבר האחרון ב-stack והיא משמשת כאשר כל הקשתות המחוברות לאיבר זה נוצרו. פעולת **NODEX**יוצרת קודקוד חדש (unit) שמשמש כאב של האלמנט הראשון ב-stack. בפעולה זו מוגדר גם התיוג של הקשת שמחברת ביניהם. הפעולה **LEFT-EDGEX** מחברת את שני האלמנטים הראשונים ב-stack, כאשר האב הוא האיבר השמאלי, ו-**RIGHT-EDGEX**עושה פעולה דומה, כאשר האב הוא האיבר הימני. בשתי הפעולות האלה נבחר גם התיוג של הקשת שנוצרה. ב-UCCA, לאיבר מסוים יכול להיות אב אחד שהוא האב העיקרי, ומספר אבות משניים (במקרה זה הוא נחשב remote participant). על מנת שלא ייווצרו גרפים לא תקינים האלגוריתם לא מאפשר שימוש ב-LEFT/RIGHT-EDGE כאשר כבר קיים לבן אב. לכן מוסיפים שתי פעולות **LEFT-REMOTEX** ו-**RIGHT-REMOTEX**על מנת לאפשר להוסיף אבות משניים. פעולת **SWAP** מאפשרת העברה של האיבר השני ב-stack ל-buffer. לבסוף, פעולת **FINISH** מסירה את השורש מה-stack ומכריזה על סיום.

המודל המרכזי במאמר הוא bi-directional LSTM. ניתן גם להשתמש ב-BERT לצורך ה-embeddings שנכנסים ל-LSTM. להלן תרשים של ה-BiLSTM:



מאמר אחר בשם Broad-Coverage Semantic Parsing as Transduction (Zhang et al., 2019) מציג גישה שונה לפתרון. בשיטה זו משתמשים בארכיטקטורת encoder-decoder. ה-encoder מקבל את המשפט, וה-decoder מקבל את המשפט ואת התחזית שלו עד לנקודה זו וחוזה את הקשת הבאה. התחזית מתבצעת בשלושה שלבים. שלב ראשון – חיזוי ה-target node, שלב שני – חיזוי האב שלו, ובשלב השלישי חיזוי הקשר ביניהם.



כעת לשאלה כיצד מבצעים אבלואציה לגרף UCCA בהינתן גרף ה-GT והגרף החזוי. נגדיר כאשר הוא האב ו- הבן, ו- הוא סט כל העלים (כזכור, העלים הם מילים במשפט המקורי) שהם צאצאים של . בנוסף, נגדיר:

קבוצת כל הזוגות של הקודקודים בשני הגרפים שהצאצאים שלהם זהים (כלומר שהתיוג של המבנה של היה נכון) ושהתיוגים של הקשתות זהים (כלומר שהתיוג של סוגי הקשתות של היה נכון). ניתן בצורה כזו להגדיר recall ו-precision:

*UCCA יכול להיות שימושי במספר תרחישים. ראשית, ניתן להשתמש בו עבור text to text generation, כגון text simplification או תרגום. בנוסף, ניתן להשתמש בו ככלי לאבלואציה של אלגוריתמים כאלה. למשל ניתן להשתמש באלגוריתם חדש לתרגום משפה לשפה, ולאחר מכן לחזות את ייצוג ה-UCCA במקור וביעד ולמדוד דמיון.*

*לסיום, קיימות עבודות אשר מנסות לאחד ייצוג*  *Distributional semantics(מבוסס וקטורים) וייצוג Symbolic Semantic Representations (מבוסס מבנים סמנטיים). המאמר Infusing Finetuning with Semantic Dependencies שנכתב על ידי Wu et al. (2021).*