שיעור 7

25 במאי 2019

1 עיבוד שפה טבעית

בשיעור זה נעסוק בעולם בעיה מסוים - עיבוד שפה טבעית, NLP. זהו תחום דעת רחב ביותר, עם שיטות מגוונות - החל בשיטות קלאסיות וכלה בשיטות מבוססות רשתות נוירונים. סוגי הבעיות עמן נרצה להתמתודד רבים, החל מזיהוי מאפיינים של טקסט (למשל, זיהוי רגשות הדובר) דרך זיהוי תיאור טקסטואלי של אובייקט וכלה במתן מענה לשאלות.
את העבודה עם שפה טבעית מאפיינים כמה אתגרים מיוחדים:

1. מידע סדרתי בסדרות ארוכות. בעיבוד שפה טבעית נרצה לנתח משפטים, פסקאות ואף מסמכים שלמים. כידוע לנו, עבודה עם מידע סדרתי מקשה על עדכון משקולות ברשתות נוירונים ומערימה קשיים גם לשיטות אחרות.

2. גודל המילון הוא רב. התמודדנו בעבר עם שאלות של feature extraction למידע קטגוריאלי. אין פתרונות טובים לבעיה זו, וככל שאוסף ה"קטגוריות" גדול יותר, כך הבעיה נעשית קשה. בשפה טבעית יש מאות אלפי מילים ללא מבנה ברור של מרחב וקטורי עליהן. שאלת ההתמודדות עם מילים רבות כל כך מצריכה שימוש בשיטות שטרם ראינו.

3. מילים הן, באופן כללי, נדירות. רוב המילים מופיעות במסמך כמות קטנה ביותר של פעמים, דבר שבאופן טבעי מקשה על למידה ומעודד over fit.

בשיעור זה נעסוק בעיקר בבעיית הייצוג של מילים - נרצה להבין כצד לתאר מילים, משפטים או פסקאות בצורה וקטורית לשימוש באלגוריתמים מוכרים. למשל, נרצה לחפש דרכים טובות לתאר כל מילה בוקטור לשימוש בשיטות שפיתחנו בעבר, כמו LSTM או GRU. באופן זה נוכל לקבל מידע טקסטואלי, לקודד אותו לשפה "וקטורית" ולהשתמש בו באופן דומה לסוגי מידע אחרים שהכרנו.

1.1 ייצוג של מילים

האתגר הגדול עמו נרצה להתמודד הוא ייצוג קומפקטי ונכון של מילים במשפט כוקטורים. מרגע שנצליח לעשות זאת נוכל להשתמש בשיטות מוכרות לנו לעבודה עם משפטים - למשל, ברשתות recurrent neural network שיקבלו סדרות וקטורים המייצגים מילים במשפט.

1.1.1 שימוש ב־1 hot encoding וב־bag of words

השיטה הנאיבית לעשות זאת היא, כזכור, שימוש ב־1 hot encoding - נבנה וקטורים באורך המילון, $|V|$, כאשר הייצוג של המילה ה־i הוא הוקטור e_i. בשיטה זו אנו מבטיחים היעדר

אינפורמציה על הקשר בין כל שתי מילים - כל שתי מילים הן מאונכות זו לזו ואין יחס סדר סמוי ביניהן. אולם בשיטה זו אנו נותרים עם ייצוג מאוד sparse, ונדרשים לוקטורים רבים כדי לתאר משפט יחיד. כלומר, הייצוג של משפט יחיד גדל במהירות ככל שיש בו יותר מילים. פתרון פשוט לכך הוא שימוש ב־bag of words - עבור כל מילה במשפט נקצה וקטור בשיטת 1 hot encoding ונחבר את כל הוקטורים הללו לוקטור יחיד, שמייצג את כל המשפט. כעת הוקטור סופר כמה פעמים הופיעה כל מילה במשפט. באופן זה עברנו לייצוג בגודל קבוע לכל משפט, פסקה או אפילו מסמך. יש בכך יתרון גדול - נוכל להשתמש באלגוריתמים סטנדרטיים רבים על וקטורים אלו. אולם הוקטורים שנותרנו איתם עודם גדולים מאוד. חסרון נוסף בשיטת ייצוג זו הוא איבוד הסדר המקורי בתוך המשפט: למשל, המשפטים "חתול שתה חלב" ו"חלב שתה חתול" יקודדו בדיוק באותו האופן, ולאלגוריתמים שנפעיל לא תהיה יכולת להבדיל ביניהם.

$$cat \mapsto \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = e_{cat}, \;\; and \mapsto e_{and}, \;\; dog \mapsto e_{dog}$$

$$cat\, and\, dog \mapsto (e_{cat} + e_{and} + e_{dog})$$

1.1.2 שימוש ב־n-gram

פעמים רבות נרצה לתת הקשר רחב יותר מזה שניתן להשיג באמצעות שימוש במילון שמתאר כל מילה בנפרד. נוכל להשתמש במילון המכיל זוגות של מילים, כל ש־(החתול, הלך) יחשב כמילה, בעוד (החתול, ישב) היא מילה אחרת. נוכל להכליל את הקונספט ל־n-grams כדי לתאר ביטויים מורכבים אף יותר. היתרון בטכניקה זו הוא שהיא מאפשרת לקודד צירופים שיש להם משמעות רק כחלק מצירוף. למשל, "New York" תוכל להיחשב במידול של 2-gram כמילה יחידה ונוכל להתייחס אליה באופן מיוחד, בעוד במודל פשוט של 1-gram לא נוכל "לראות" אותה.

1.1.3 TF-IDF

בקידוד bag of words אנו לא מניחים כל ידע א־פריורי על המילים שאנו מקודדים. ישנה אינפורמציה שנוכל לגלות על מילים עוד בטרם הפעלנו קידוד כלשהו: למשל, נוכל להבין אילו מילים הן נפוצות, או אילו מילים מאפיינות נושא מסוים. קידוד TF-IDF נעשה בהקשר של קורפוס, אוסף גדול של מסמכים. נרצה לקודד כך מסמך לפי המילים שמופיעות בו, כאשר ניתן לכל מילה משקל בהתאם למידה בה היא "מאפיינת" את המסמך. ניתן היה לחשוב כי ככל שמילה מופיעה יותר במסמך, כך היא יותר מאפיינת אותו. אולם, למשל, המילה האנגלית and מופיעה פעמים רבות בכל אחד מדפי הויקיפדיה באנגלית. אולם אין פירוש הדבר שיש בה כדי ללמוד מידע משמעותי על ערך ספציפי בו היא מופיעה: זאת כיוון שהיא מופיעה גם בכל הערכים האחרים. לכן נרצה להתחשב בה פחות אם היא מופיעה בערכים רבים. כלומר, נרצה להתחשב פחות במילים שמופיעות בערכים רבים. לשם־כך נרצה לשקלל בין שני הגדלים.

נשתמש אם־כן באבחנה לפיה אם מילה x היא מילה נדירה בשפה באופן כללי, אך מופיעה

פעמים רבות במסמך ספציפי - פירוש הדבר שמילה זו קשורה באופן כלשהו לנושא המסמך. בקידוד TF-IDF אנו מגדירים, עבור כל מסמך ולכל מילה, ציוני tf ו־idf. ציון tf - term frequency, הוא תדירות ההופעות של מילה בכל המסמכים בקורפוס. נסמן ב־x מילה בודדת, ב־d מסמך (המכיל מילים) וב־D או אוסף המסמכים שלנו. נגדיר אותו ע"י

$$tf(x,d) = |\{w \in d | w = x\}|$$

כלומר, כמות ההופעות של המילה x במסמך d.
ציון idf - inverse document frequency הוא מדד לכמה אינפורמציה המילה מספקת. נגדיר אותו ע"י

$$idf(x,D) = log\frac{|D|}{|\{d \in D | x \in d\}|}$$

כאשר D הוא אוסף המסמכים, הקורפוס. כלומר, אנו שואלים באיזה חלק מהמסמכים מופיעה המילה x, ולוקחים את ההופכי למספר זה. ננרמל את הגודל באמצעות לקיחת לוגריתם. למעשה, יש שיטות רבות לנרמל את ציוני ה־tf וה־idf. ציון ה־TF-IDF הסופי יקבע ע"י

$$tfidf(x,d,D) = tf(x,d) \cdot idf(x,D) = |\{w \in d | w = x\}| \cdot log\frac{|D|}{|\{d \in D | x \in d\}|}$$

הציון שקיבלנו הוא, כאמור, עבור כל מילה ועבור כל מסמך. נוכל לתאר כל מסמך באמצעות הציון שמקבלת בו כל מילה. נראה לכך דוגמא: למשל, עבור אוסף המסמכים שהוא אוסף המשפטים הבא:

$$\{"he\,walks\,here", "she\,walks\,here"\}$$

נקבל למשל כי

$$tfidf("he", 1) = 1 \cdot log\frac{2}{1} \simeq 0.70$$

נקבל עבור על מסמך וקטור באורך אוסף המילים האפשרי, או המילון. פעמים רבות נרצה לנרמל אותו כך שנקבל לכל מסמך וקטור באורך 1. השימוש ב־TF-IDF הוא יעיל, ונוכל להשתמש בו כדי לאפיין משפטים, פסקאות או מסמכים שלמים ולהשוות ביניהם. בהינתן שני מסמכים נוכל לחשב, למשל, את המרחק בין וקטורי ה־TF-IDF שלהם כדי לקבל מושג למידת הדמיון שלהם. זהו שימוש נוסף שלא יכולנו לעשות כלל באמצעות 1 hot encoding למילים בודדות (אף שיכולנו לעשות אותו עבור bag of words).

1.2 Word2Vec

נדון כעת במודל מפתיע בפשטותו שפורסם ע"י google בשנת 2013. מטרת המודל היא, כאמור, למצוא ייצוג לכל מילה במילון. המודל מגיע מתוך האבחנה הבאה. נתבונן במשפט הבא עם מילה חסרה:

$$once\,upon\,a\,\_\_\_\,in\,a\,land\,far\,away$$

נוכל בקלות להבין, מתוך ההיכרות שלנו עם השפה האנגלית, כי המילה החסרה היא $time$. כלומר, אנו מסוגלים לאפיין מילה באמצעות הקונטסט, הסביבה בה היא מופיעה. במידול

של word2vec נוסיף לאמירה הזו, ונטען כי ניתן לאפיין כל מילה באמצעות התפלגות על כל המילים במילון, שקובעת אילו מילים סביר שנמצא בסביבה שלה. נטען כי ישנו קשר חד־חד ערכי בין מילים לבין ההתפלגויות הללו, ונעשה בקשר זה שימוש כדי ללמוד ייצוג דחוס של מילים.

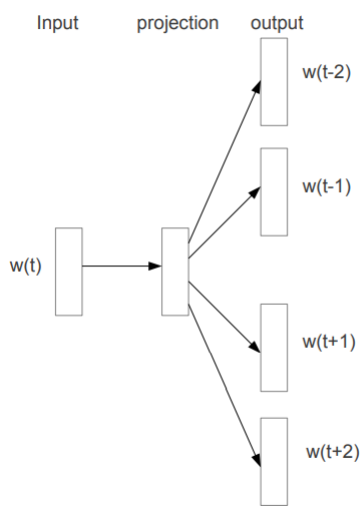
נפתח כעת דיון במודל word2vec מסוג skip-gram. ניצור את הייצוג באופן הבא: נתחיל מייצוג של מילים ב־1 hot encoding. המודל שלנו יקבל מקורות טקסט רבים. המודל המקורי אומן על אוסף מאמרים עצום של google news. נרצה, עבור כל מילה, לחזור את המילים שסביבה. כלומר, עבור המילה $w(t)$ נרצה לחזות, למשל, את

$$w(t-2), w(t-1), w(t+1), w(t+2)$$

בפועל נעבור על כמות טקסט רבה וניצור סט אימון המורכב מהזוגות

$$(w(t), (w(t-2), w(t-1), w(t+1), w(t+2)))$$

כעת נבנה רשת נוירונים פשוטה:



איור 1:

כאשר גודל שכבת הקלט הוא כגודל המילון, גודל שכבת הפלט הוא בגודל כמה עותקים של גודל המילון וגודל השכבה ביניהן הוא גודל הקידוד שלנו, אותו נסמן ב־d. נהוג לקבוע, עבור בעיית word2vec כללית, $d \sim 300$. באופן מפתיע, ניתן לאמן את הרשת ללא אקטיבציה על השכבה הפנימית. עם זאת, על השכבה החיצונית נוסיף אקטיבציה של soft max. אקטיבציה זו תשמש אותנו ליצור התפלגות על פני כל הפלטים האפשריים, והיא נתונה ע"י

$$x_j \mapsto \frac{e^{x_j}}{\sum_i e^{x_i}}$$

באופן זה אנו מקבלים פונקציית אקטיבציה גזירה שמחזירה התפלגות על אוסף הפלטים, כך שסכום ההסתברויות הוא 1.

נוכל לאמן את המודל באמצעות פונקציית loss כמו $binary\,crossentropy$ שפגשנו בעבר:

$$-(y \cdot log(p) + (1-y) \cdot log(1-p))$$

המודדת מרחק בין התפלגויות.
בהינתן מודל מאומן, נתבונן במטריצת המשקולות שהתקבלה במעבר משכבת הקלט לשכבה הבאה: W_{ij}. מטריצה זו מתאימה לוקחת את המילה ה־i, המיוצגת ע"י הוקטור e_i, ומחזירה את $W_{ij}e_i$, כלומר את $(W_{ij})_{j=1}^{d}$, וקטור בגודל d המייצג את המילה ה־i. באופן זה אנו משיגים ייצוג שאינו sparse של כל המילים במילון. ייצוג זה מקיים תכונה חזקה במיוחד: מילים בעלות משמעות סמנטית דומה יקבלו וקטורים דומים במרחב השיכון. זאת כיוון שמילים בעלות משמעות דומה מופיעות בהקשרים דומים, כך ימופו לשיכונים דומים דרך השכבה הראשונה. בפרט נקבל את התכונה הרצויה הבאה: אם מילה a דומה סמנטית למילה b ולא דומה סמנטית למילה c, אזי $d(v_a, v_b) < d(v_a, v_c)$ עבור v_i השיכון הנלמד של המילה ה־i. תכונה בסיסית זו היא חשובה ביותר, למשל להפעלת אלגוריתמי clustering.
אלגוריתם word2vec מפגין עוד תכונה מעניינת ביותר: במרחב השיכון שהוא יוצר מתקיימות תכונות ליניאריות לא טריוויאליות. נמדוד מרחקים בתוך המרחב המשוכן באמצעות מטריקה של cosine similarity, המוגדרת ע"י

$$d(v_1, v_2) = 1 - \frac{< v_1, v_2 >}{||v_1|| \cdot ||v_2||}$$

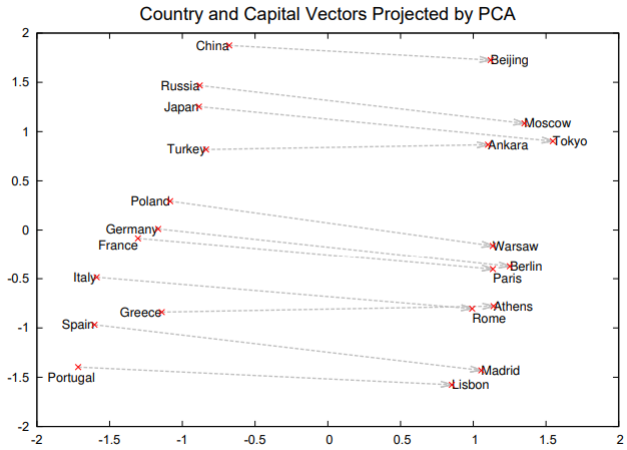
מטריקה זו מודדדת את הזווית בין שני וקטורים. במחקר של אלגוריתם word2vec התגלו קשרים מהסוג הבא:

$$v_{king} - v_{queen} \approx v_{man} - v_{woman}$$
$$v_{france} - v_{paris} \approx v_{germany} - v_{berlin}$$

וכן

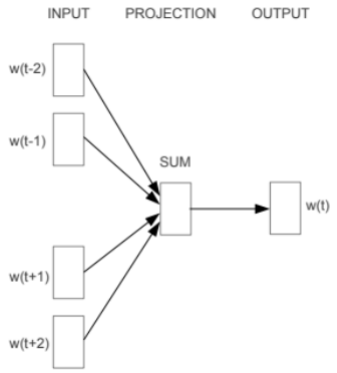
$$v_{russia} + v_{river} \approx v_{volga\,river}$$
$$v_{germany} + v_{capital} \approx v_{berlin}$$

כלומר, נראה כי ניתן לחבר ולחסר וקטורים של מילים לקבלת אנלוגיות בשפה ולקבל קשרים פשוטים בין מילים ומושגים. כלומר, במובן מסוים, ישנו וקטור יחיד המקשר בין "מדינות" לבין "ערי בירה", או בין "זכר" ו"נקבה". קשר זה אינו מוסבר היטב אף בספרות, אך מעניין חוקרים רבים.



איור 2:

כיצד נשתמש ב־word2vec? כאמור, ייצוג דחוס ואינפורמטיבי של מילים הוא שימושי ביותר: נוכל להשתמש בו כדי להפעיל אלגוריתמים שלמדנו להכיר בשבועות האחרונים, החל מ־clustering וכלה באלגוריתמי חיזוי ואף ייצור טקסט. כזכור, עסקנו מעט ב־transfer learning: שיטה בה אנו לוקחים רשת מאומנת ומשתמשים במשקולות שלה כדי להתאים אותה לבעיה דומה. נוכל לעשות שימוש ב־word2vec באופן דומה: עבור כל בעיית טקסט נוכל ליצור רשת שמקבלת מילים כוקטורי 1 hot encoding ולהעביר אותם דרך שכבה ליניארית, בעלת משקולות קפואות שיאותחלו למשקולות של מודל word2vec. באופן זה נוכל להשתמש בקלות בייצוג word2vec שנלמד על כמויות טקסט גדולות במשימות שונות.
נעיר כי את הדיון פיתחנו עבור מודל מסוג skip-gram. ישנו מודל נוסף שנעדיף להתבונן בו בהמשך - CBOW (Continuous Bag of Words). הוא דומה מאוד למודל ה־skip-gram ומתואר ע"י הארכיטקטורה הבאה:



איור 3:

באמצעותו אנו חוזים, בהינתן קונטקסט, את המילה המתאימה. כלומר, אנו מקבלים את המילים בסביבת המילה אותה אנו מעוניינים לחזות, מתבוננים ב־embedding שלהן, מחברים אותם וחוזים את המילה המתאימה.

1.2.1 Negative sampling

נציג כעת שיטה נפוצה לאימון embedding באופן כללי. למעשה, נעשה בשיטה זו שימוש גם לאימון המקורי של word2vec. אלגוריתמי embedding רבים ניתן לאמן דרך הפורמליזם הבא:
נניח כי f היא פוקנציית ה־embedding, $f : V \to E$, כאשר f תלויה במשקולות θ. לכן נכתוב $f(v;\theta)$. אנו מעוניינים לקיים את הכלל לפיו דוגמאות "דומות" ישוכנו קרוב אחת לשנייה. לכן נוכל להגדיר את האוסף

$$B = \{(u,v) \in V \times V | u \approx v\}$$

כלומר, אוסף הזוגות של איברים דומים. המטרה שלנו היא לקרב בין איברים דומים, כך שה־loss יוכל להיות

$$L = \sum_{(u_i,v_i)\in B} d(f(u_i;\theta), f(v_i;\theta))$$

כאשר d היא המטריקה במרחב השיכון, E. נשים לב כי לאו דווקא מתקיים כי f היא רשת נוירונים - כל שנדרוש הוא פונקציה שנוכל לבצע באמצעותה אופטימיזציה על E. למשל, בהינתן שהמטריקה גזירה, נוכל לדרוש שגם f תהיה גזירה לקבלת פונקציה כנדרש.
כעת נוכל לדרוש גם כי איברים שאינם דומים יהיו בעלי embedding רחוקים אלו מאלו. נעשה זאת ע"י הגדרת הקבוצה

$$C = \{(u,v) \in V \times V | u \not\approx v\}$$

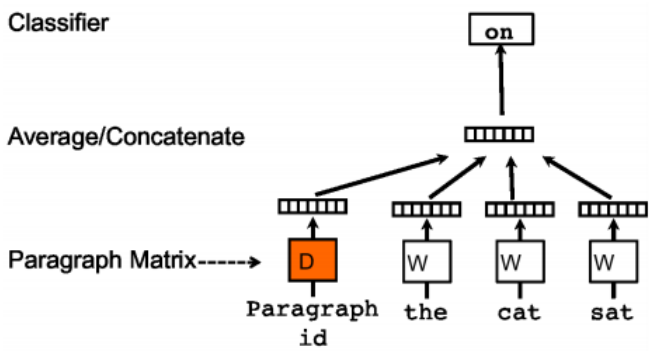
והוספת רכיב ל־Loss:

$$L = \sum_{(u_i,v_i)\in B} d(f(u_i;\theta), f(v_i;\theta)) - \sum_{(u_j,v_j)\in C} d(f(u_j;\theta), f(v_j;\theta))$$

באופן זה נוכל לעודד למידה של embedding עם התכונות הרצויות לנו. שיטה זו, בה אנו דוגמים זוגות שלא נרצה שיהיו דומים, נקרא negative sampleing. לרוב נדגום יותר דוגמאות שליליות מחיוביות לחישוב הסופי של ה־Loss. כאמור, בחישוב word2vec פעמים רבות משתמשים בשיטה זו.

1.3 Doc2Vec

נרצה להיות מסוגלים ליצור embedding, קידוד, גם למשפט, או אף למסמך שלם, ולא רק למילים בודדות. מהו הפתרון הנאיבי לקידוד משפט, בהינתן שבנינו מודל word2vec? נוכל, כפי שהכללנו את 1 hot encoding ל־bag of words, להתבונן בממוצע או בסכום של הוקטורים המייצגים את המילים במשפט לקבלת ייצוג של המשפט עצמו. זהו פתרון לא רע, אך פשוט למדי. נוכל לחפש פתרון מתוחכם יותר, שיוכל "לתפוס את ההקשר" של מסמך באופן טוב יותר. בהינתן אוסף מסמכים $\{D_i\}$ נוכל לעשות זאת בפשטות, ע"י כך שנאמן מודל דומה מאוד למודל ה־CBOW:



איור 4:

נאמן את המודל באותו האופן בדיוק למודל ה־CBOW, אלא שנוסיף לכל קונטקסט שאנו מכניסים לרשת וקטור 1 hot encoding שמתאר את מספר המסמך ממנו לקחנו את חלון המילים עליו אנו עובדים. בכך נאפשר לרשת ללמוד להשתמש במידע הנוגע למקור ממנו נלקחו המילים כדי לשכן אותן בצורה טובה יותר. נקבל בסופו של דבר וקטור embedding לכל מסמך בנפרד. נוכל להשתמש כעת בוקטורים אלו כדי לתאר מסמכים שלמים, ללא תלות בכמות המילים שיש בהן. שימוש מעשי ל־doc2vec הוא, למשל, שיכון של משפטים.
נדגים: נניח שאנו רוצים לאמן מודל doc2vec עבור שלושת המשפטים הבאים:

1. החתול הלך לגן.
2. הילד הלך לגן.
3. הילד מלטף את החתול.

נקצה לכל משפט מספר. כעת נוכל להקצות לכל משפט קידוד 1 hot encoding המתאים למספר. נכניס את הקידוד לרשת בתהליך האימון שלה, כפי שמתואר בארכיטקטורה לעיל. מתהליך האימון נקבל כי וקטור ה־1 hor encoding המייצג את המשפט עובר לוקטור מרחב השיכון. וקטור זה מייצג את המשפט, והוא אומן באמצעות הניסיון לחזות מילים במשפט. לכן הוקטור המתקבל "מייצג" במידה כלשהי את המשפט, או לכל הפחות רלוונטי לעזרה בתהליכי חיזוי דומים עליו.

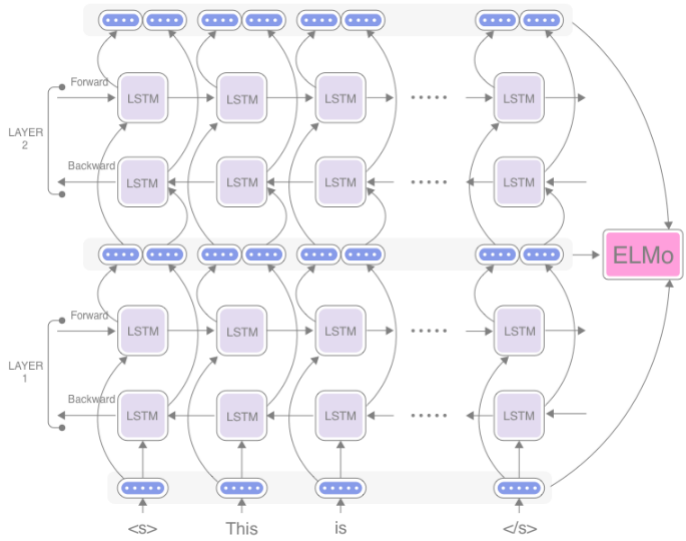
1.4 מודל שפה ו־Elmo

מודל שפה הוא מודל אשר, בהינתן כמה מילים במשפט, חוזה את המילה הבאה בו. ניתן להשתמש במודלים שכאלו כדי ליצור embedding איכותי למילים בהינתן קונטקסט. למשל, במודל Elmo (Embeddings from language models) עושים שימוש במודל מאומן שכזה ומתאימים אותו לבעיה מסויימת. נזכיר כי בשיעור הקודם עסקנו בהרחבה במודלי שפה. בפרט, ב־Elmo נשתמש במודל שפה מאומן המורכב מכמה שכבות biLSTM המורכבות זו על זו. מהמעבר ב־LSTM בכל שכבה נוכל לקבל וקטור קונטקסט, h_i ל־$i = 0, ..., n$ כאשר $i = 0$ היא שכבת הקלט ו־i_n הוא השכבה האחרונה במודל. זהו ה־hidden של כל שכבה ברשת, ומקודד, כזכור, "זיכרון" של הרשת על הקלט שעבר בה.
נוכל לחשוב על אוסף הוקטורים h_i כעל קידוד כללי למילים בהקשר מסוים, או כעל קידוד למשפט. אולם נרצה להתאים את הקידוד שלנו למשימה מסוימת. למשל, נרצה לחזות

האם משפט מסוים מביע רגע חיובי או שלילי. נוכל לאמן מסווג פשוט: הוא יקח את אוסף הוקטורים h_i, יחבר ביניהם באמצעות ממוצע משוקלל כלשהו (כאשר הוא ילמד את המשקלים) וישתמש בוקטור ה"ממוצע" לחיזוי. לכן נקבל בתהליך ביטוי מהצורה הבאה:

$$\sum_i s_i h_i$$

כאשר s_i הם משקול של כל ייצוג של הטקסט שהכנסנו למודל השפה ברמות שונות של המודל. הדבר מאפשר לתפוס ייצוגים של המודל בכמה רמות: ברמה הראשונה מעובדים ייצוגים בסיסיים של הטקסט, הנוגעים יותר ל־syntax ולמבנה המילים. בשכבות האחרונות נלמדים קישורים סמנטיים רחבים יותר. נוכל לאמן את המודל לקבלת הפרמטרים s_i בהינתן בעיה ספציפית שנרצה להתמחות בה. למשל, לבעיה של הבנת שורש המילה נלמד להשתמש בייצוגים "נמוכים" של משפט, בעוד לבעיות מסובכות יותר אנו עשויים להידרש למשקל גדול יותר לייצוגים "גבוהים" של המשפט.
נעיר כי ישנן כמה דרכים לקבל את s_0: לרוב s_0 הוא העברה של המשפט המקורי דרך שכבה כלשהי (למשל שכבת LSTM או אף שכבת קונבולוציה).



איור 5: