סיכום NLP

26 בפברואר 2023

1 מודלי שפה:

1.1 בעיות של מודלים:

- **דלילות - $sparsity$:** נאמר שהמידע הוא דליל אם אין לנו מספיק דוגמאות על מילה מסויימת בכדי לדעת איך משתמשים בה.
  מכיוון שמילים מתפלגות בהתפלגות זיפ, אזי תמיד יהיו מילים שיופיעו מספר מועט של פעמים.

1.2 מודלים:

- **המשימה:** בהינתן מילה נרצה לחזות את המילה הבאה. או במילים אחרות - לשערך את ההתפלגות לקבל את הסטרינג הסופי הבא בהינתן מספר סופי של סטרינגים. בנוסף למילים יש סטרינג שמסמל התחלה וסטרינג שמסמן סוף משפט.

$$\sum_{x \in \mathcal{V}^*} p(x) = 1$$

- **מודל להמרת אודיו לטקסט:** מודל זה יעריך את ההסתברות לכך שאדם יקליט את המשפט שהמערכת שמעה, והמערכת תבחר את המשפט שיש לו את ההסתברות הגבוהה ביותר להיאמר.
  נבנה מודל שאומר בהינתן משפט כיצד סביר שייראה הסיגנל האקוסטי שלו. ומודל נוסף שיגיד לנו מה סביר שיהיה.

- **דרגת ניתנות לחיזוי של שפה (קלוד שאנון):** שאנון חקר מה ההסתברות להופעת המילה הבאה בטקסט. וניתן לדרג שפה כמה היא צפויה ומה ההסתברות להצליח לנחש את האות הבאה, בהינתן האות שלפניה.

- **מה המודל צריך לקיים:**
  1: המודל צריך להיות מודל התפלגות - סכום הסתברות הסטרינגים = 1.
  2: המודל יחזה משפטים שיותר סביר שיאמרו.

- **מודל שפה טרוויאלי:** ההסבתרות של סטרינג היא מספר הפעמים שהוא הופיע בסט האימון חלקי גודל סט האימון.
חסרון: המודל לא מכליל, ונותן 0 לכל משפט שלא הופיע בסט האימון - **סובל מדלילות**.

$$p\left(x_1 \ldots x_n\right) = \frac{c\left(x_1 \ldots x_n\right)}{N} \text{ for sentence } x = x_1 \ldots x_n$$

- **מודל** $Unigram$**:** מודל זה משתמש בהנחה כי ההסתברות לקבל משפט כלשהו שווה למכפלת ההסתברויות של המילים שמרכיבות אותו. במילים אחרות - כל מילה ב"ת בשאר המילים.
מודל זה **לא** סובל מדלילות נתונים, אך הוא לא שימושי מכיוון שהוא מניח אי תלות שלא נכונה בעולם האמיתי.

$$p\left(x_1 \ldots x_n\right) = \prod_{i=1}^{n} p\left(x_i\right)$$

- **מודל** $Bigram$**:** מודל מרקובי מסדר ראשון. הנחת המודל היא כי - ההסתברות של סטרינג x_i היא בלתי תלויה בכל $x_1 ... x_{i-2}$ בהינתן x_{i-1}. כלומר כל סטרינג (מילה) תלוי רק בסטינג שמופיע לפניו.

$$p\left(x_i \mid x_{i-1} \ldots x_0\right) = \prod_{i=1}^{n} p\left(x_i \mid x_{i-1}\right)$$

כאן הסיבוכיות היא V^2 עבור משפט באורך V (מספר המילים במשפט).

- **מודלים מרקובים:** יש לנו אוטומט עם מצבים וקשתות. מכל מצב יש קשתות עם משקל המייצג את ההסתברות לקבל את המצב הבא, והמצב הבא תלוי רק במצב שאנו נמצאים בו כעת.
ההסתבורת הכוללת לקבלת המשפט שווה למכפלה על הסתברות המעברים בין המילים.

$$p\left(x_1 \ldots x_n\right) = \prod_{i=1}^{n} p\left(x_i \mid x_{i-1}\right)$$

ניתן לייצג את המודל המרקובי כמטריצת מעברים, עם כל המילים על השורות ועל העמודות, ובכל תא i, j תופיע ההסתברות לעבור מהמילה ה i למילה ה j.

- **מודל מרקוב מסדר גבוה:** במקום לבחור את המילה הבאה בהסתמך על המילה הנוכחית בלבד. אנו נסתכל על שתי המילים או שלש המילים האחרונות וכך נבחר את המילה הבאה.
עבור מודל מסדר j:

$$p\left(x_1 \ldots x_n\right) = \prod_{i=1}^{n} p\left(x_i \mid x_{i-1} ... x_{i-j}\right)$$

ניתן לצייר אותו גם כאוטומט, אך כאן כל מודל ייוצג עם הסתברות של המילים שלפניו.

1.3 למידת מודל $Bigram$:

- **כדי ללמוד מודל** $Bigram$ מסדר j, נחשב את פונקציית הצפיפות המצטברת, כלומר הסיכוי של המילה ה i להופיע אחרי j המילים שהיו לפניה.

נחשב כך: נחלק את מספר הפעמים שהמילה שבחרנו מופיעה אחרי רצף המילים, חלקי מספר כל המילים שהופיעו אחרי רצף המילים.

$$q_{ML}\left(x_m = w_j \mid x_{m-1} = w_i\right) = \frac{count\left(w_i, w_j\right)}{\sum_{j'} count\left(w_i, w_{j'}\right)}$$

- **החסרון:** אם מילה מסויימת לא הופיע בסט האימון $count\left(w_i, w_j\right) = 0$, נקבל הסתברות אפס למילה הזאת, על אף שהמשפט הגיוני ונכון.

- **מה נרצה:** נרצה שההסתברות לא תהיה 0, אלא נמוכה וקרובה ל 0. משום שאנו מודעים לכך שהדאטה לא מושלם. שיטות אלו נקראות שיטות החלקה.

1.4 שיטות החלקה:

- **הרעיון:** אנו מודעים לכך שאין לנו דאטה של על המשפטים בעולם. ולכן מילה שהופיעה פעם אחת לא תהיה טובה יותר בהרבה מאשר מילה שלא הופיעה אף פעם.
לכן אף פעם לא ניתן אפסים.

1.4.1 שיטת הוספת דלתא - δ:

- **הרעיון:** נוסיף לכל קבוצת מילים δ, כך שבמקום k הופעות יהיו להן $k + \delta$ הופעות, אך לאחר מכן ננרמל במספר המילים + מספר ה δ-ות שהוספנו.

$$q_{add-\delta}(w) = \frac{c(w) + \delta}{\sum_{w'}\left(c\left(w'\right) + \delta\right)} = \frac{c(w) + \delta}{c() + \delta|\mathcal{V}|}$$

באופן זה נחלק בפרמטר גדול יותר, וכל מילה שההסתברות שלה להופיע הייתה גבוה, ההסתברות שלה להופיעה תרד. ומילים שההופעה שלהן הייתה - 0 יקבלו הסתברות קטנה להופעה.

- **חסרון:** זה לא עובד. יש מספר בעיות, לדוגמה כאשר מילה מופיעה מספר מועט של פעמים ההסתברות מצטברת שמתקבלת לאחר הדלתא לא הגיונית.

1.4.2 שיטת $Back-off\ Models\ (Linear\ Interpolation)$:

- **הרעיון:** ננסה לשלם בין כמה מודלים ע"י קומבינציה קמורה, נשלב בין $Unigram, Bigram, Trigram$.

נעשה זאת כך: נתחיל עם להסתכל על מודל $Trigram$ ונחשב אותו. אם אין מספיק מילים - נעבור להסתכל על

מודל $Bigram$, ואם גם למודל זה אין מספיק אפשרויות - נסתכל על אחוז ההופעה של המילה מכלל מאגר המילים.

$$\begin{aligned} q\left(w_i \mid w_{i-2}, w_{i-1}\right) &= \lambda_1 \times q_{ML}\left(w_i \mid w_{i-2}, w_{i-1}\right) \\ &+ \lambda_2 \times q_{ML}\left(w_i \mid w_{i-1}\right) \\ &+ \lambda_3 \times q_{ML}\left(w_i\right) \end{aligned}$$

עבור $\lambda_1 + \lambda_2 + \lambda_3 = 1$ וגם $\lambda_i \geq 0$ לכל λ.

- **כיצד נבחר את הלמדות:** נשאיר בצד חלק מסט האימון, ונבדוק על החלק הזה מה הלמדה שמחזירה לנו את ההסתברות הטובה ביותר למשפט.

- **מקסום הנראות:** בהינתן סט, נבדוק מה פונקציית ההתפלגות ממנה הוא נדגם.
פונקציית ההתפלגות המצטברת שלנו היא:

$$L\left(\lambda_1, \lambda_2, \lambda_3\right) = \log\left(\prod_{(w_1,w_2,w_3)\in V^3} q\left(w_3 \mid w_1, w_2\right)^{c'(w_1,w_2,w_3)}\right) = \sum_{w_1,w_2,w_3} c'\left(w_1, w_2, w_3\right) \log q\left(w_3 \mid w_1, w_2\right)$$

כאשר $c'\left(w_1, w_2, w_3\right)$ הוא מספר הפעמים ש $Trigram$ - $\left(w_1, w_2, w_3\right)$ מופיע בסט האימון. ו q מייצג את ההסתברות.

- **אופציה נוספת לבחירת λ-ות:** נבחר מספר למדות כל שכל אחת מייצגת מקרה אחר.

  - It is common to take more ls to get a better fit
  - Here is one example of how to do that. Define:

$$\Pi(w_{i-2}, w_{i-1}) = \begin{cases} 1 & \text{If Count}(w_{i-1}, w_{i-2}) = 0 \\ 2 & \text{If } 1 \leq \text{Count}(w_{i-1}, w_{i-2}) \leq 2 \\ 3 & \text{If } 3 \leq \text{Count}(w_{i-1}, w_{i-2}) \leq 5 \\ 4 & \text{Otherwise} \end{cases}$$

  - And maximize:

$$\begin{aligned} q(w_i \mid w_{i-2}, w_{i-1}) = \; & \lambda_1^{\Pi(w_{i-2},w_{i-1})} \times q_{\text{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\ & + \lambda_2^{\Pi(w_{i-2},w_{i-1})} \times q_{\text{ML}}(w_i \mid w_{i-1}) \\ & + \lambda_3^{\Pi(w_{i-2},w_{i-1})} \times q_{\text{ML}}(w_i) \end{aligned}$$

  where $\lambda_1^{\Pi(w_{i-2},w_{i-1})} + \lambda_2^{\Pi(w_{i-2},w_{i-1})} + \lambda_3^{\Pi(w_{i-2},w_{i-1})} = 1$,
  and $\lambda_i^{\Pi(w_{i-2},w_{i-1})} \geq 0$ for all i.

1.4.3 שיטות $Discounting$:

- **הרעיון:** נבחר פקטור הורדה d ונחסר אותו מכל ההופעות של המילים (הפוך ממודל δ), ולאחר מכן ננרמל במספר המקורי של כל המילים, (זאת לא התפלגות כי היא לא נסכמת ל 1).
עבור מודל $Bigram$:

$$\lambda\left(w_{i-1}\right) = 1 - \sum_{w} \frac{c^*\left(w_{i-1}, w\right)}{c\left(w_{i-1}\right)}$$

- **שיטת** $Kneser - Ney\ Smoothing$:

$$P_{\text{CONTINUATION}}(w) \propto \left|\left\{w_{i-1} : c\left(w_{i-1}, w\right) > 0\right\}\right|$$

$$P_{\text{CONTINUATION}}(w) = \frac{\left|\left\{w_{i-1} : c\left(w_{i-1}, w\right) > 0\right\}\right|}{\left\{\left(w_{j-1}, w_j\right) : c\left(w_{j-1}, w_j\right) > 0\right\}\ |}$$

$$P_{KN}\left(w_i \mid w_{i-1}\right)=\frac{\max\left(c\left(w_{i-1}, w_i\right)-d, 0\right)}{c\left(w_{i-1}\right)}+\lambda\left(w_{i-1}\right) P_{\text{CONTINUATION}}\left(w_i\right)$$

λ הוא פרמטר נורמליזציה, ונניח כי $d \geq 1$.

$$\lambda\left(w_{i-1}\right)=\frac{d}{c\left(w_{i-1}\right)}\left|\left\{w: c\left(w_{i-1}, w\right)>0\right\}\right|$$

הרעיון: נבדוק את ההסתברות של מילה להופיע במודל $Bigram$ וגם את ההופעה של המילה בכללי. ננקד כל מילה לפי מספר ההופעות שלה לבד, ועם הופעות עם המילה שלפניה במשפט שננסה להרכיב.

1.5 מודלים כלליים:

1.5.1 המודל של $Pereira$:

- **הרעיון:** נמדל את המילים לקטגוריות c_i, ובהינתן מילה נרצה לדעת לאיזו קטגוריה c_i היא שייכת. כך נוכל לחזות את הקטגוריה של המילה ה i בהינתן הקטגוריה של המילה ה $i-1$.

$$\{Pr\left(w_i \mid w_{i-1}\right)=\sum_{c \in CATS}\{Pr\left(w_i, c \mid w_{i-1}\right)=\sum_{c \in CATS}\{Pr\left(c \mid w_{i-1}\right) \cdot\{Pr\left(w_i \mid c, w_{i-1}\right)=$$
$$\sum_{c \in CATS}\{Pr\left(c \mid w_{i-1}\right) \cdot\{Pr\left(w_i \mid c\right)$$

כעת מודל זה ממיין מילים לפי קטגוריות שיופיעו באותן הקשרים יחד. והוא ימיין משפטים בצורה טובה יותר.

1.6 הערכת מודל:

- **כיצד נעריך את טיבו של המודל:** נעשה זאת באמצעות $Perplexity$ ובאמצעות פונקציית הצטברות:
עבור $s_1 ... s_m$ משפטים

$$\text{Perplexity } =e^{-l}, \quad for \quad l=\frac{1}{M} \sum_{i=1}^{m} \log P\left(s_i\right)=\frac{1}{M} \sum_{i=1}^{m} \sum_j \log P\left(x_j^{(i)} \mid x_1^{(i)}, \ldots, x_{j-1}^{(i)}\right)$$

כאשר M הוא מספר המילים בסט הבוחן. ו m הוא מספר המשפטים בסט הבוחן.
ואנו נגיד שמודל הוא טוב אם ה $Perplexity$ שלו נמוך יותר.
החסרון: זה מאוד משתנה לפי הדאטה, ולכן הוא לא מעריך בצורה המיטבית.

2 $Classification, Log-linear\ Models$:

2.1 $Classification$:

- **למידה מפוקחת:** למידה מסט אימון שיש עליו תוויות.
- **בעיות סיווג:** יש לנו כמה קטגוריות נתונון, ובהינתן אינפוט אנחנו צריכים להחליט לאיזו קטגוריה הוא שייך.

2.2 $Probabilistic\ Classification$:

הרעיון: נרצה למצוא מסווג כך שהוא יתן לנו התפלגות על המרחב.

2.2.1 מודלים $Joint/Generative$:

מודלים שינסו לחזות עבור כל נקודה במרחב את ההסתברות לקבל את הנקודה בהינתן הקטגוריה, עבור כל הקטגוריות הקימות. כלומר איזו קטגוריה הכי מתאימה לאינפוט.
נחזה באופן הבא: נחפש את הקטגוריה y שתביא לנו את ההסתברות המקסימלית.

$$y^* = argmax_y P(y, x) = argmax_y P(y)P(x|y)$$

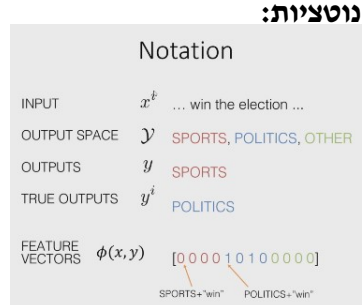
כאשר y^* זאת הפרדיקציה שלנו.

2.3 מודלים דיסקרימינטיבים $Conditional/Discriminative$:

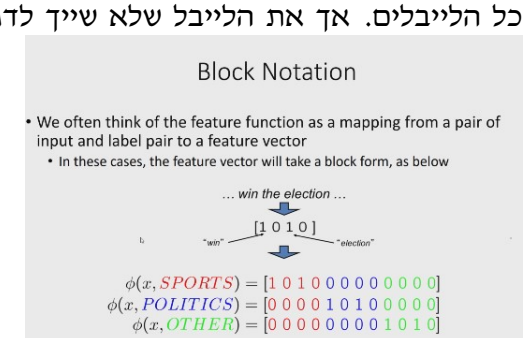
מודל שממדל את ההסתברות של y בהינתן x. הוא לא נותן חיזוי, אלא הוא משערך עד כמה הדוגמה שייכת לקטגוריה y.

$$y^* = argmax_y P(y \mid x)$$

- **ייצוג הפיצ'רים:** במודל דסקרמינטיבי ניתן לכל קטע טקסט פיצ'ר עבור הימצאות של מילים מסויימות ועוד, כדי שנוכל לסווג אותם לקטגוריה המתאימה.
- **נוטציות:**



- **בלוק נוטציות:** כשנרצה לייצג ווקטור שמתאים לכל קומבינציה של x, y, נעשה זאת ע"י שכפול של כל הדגימות עבור כל הלייבלים. אך את הלייבל שלא שייך לדגימה נמלא באפסים.



- **אומד נראות מקסימלית** - $Maximum\ Likelihood\ Estimation\ (MLE)$:

בהינתן סט אימון, נרצה לבחור את ההתפלגות ממנה נדגם הסט. נחשב את פונקציית ההתפלגות המצטברת ונבחר את האומד המקסימלי.

$$\theta_{MLE} = argmax_{\theta} P(x|y;\theta)$$

כאשר θ הם הפרמטרים של המודל.

2.3.1 מודל בייס:

אנו מניחים כי הדאטה נדגם מהתפלגות בייסיאנית, והוא מוגדר כך:

$$P(x,y) = \mathrm{P(y)} \prod_{j=1}^{d} P\left(x^{(j)} \mid y\right)$$

וה - $MLE(\hat{p})$ של המודל הוא:

$$\hat{p}(y) = \frac{\#\{y_i = y\}}{N}; \quad \hat{p}\left(x^{(j)} \mid y\right) = \frac{\#\left\{x_i^{(j)} = x, y_i = y\right\}}{\#\{y_i = y\}}$$

המודל מניח אי תלות בין הפיצ'רים של הדגימה בהינתן y.

החיזוי:

$$y^* = \{argmax_y P(y)P(x \mid y) = \{argmax_y \mathrm{P(y)} \prod_{j=1}^{d} P\left(x^{(j)} \mid y\right) =$$

$$= \{argmax_y P(y) \prod_{j:\ \textbf{word in } x} P\left(x^{(\textbf{word })} = 1 \mid y\right) \prod_{j:\ \textbf{word not in } x} P\left(x^{(\textbf{word }^2)} = 0 \mid y\right)$$

2.3.2 מודל לוג לינארי - $log\ linear\ model$:

- **המודל:** נשתמש ב $score(x,y,w)$:

$$p(y \mid x; w) = \frac{\exp(w \cdot \phi(x,y))}{\sum_{y'} \exp\left(w \cdot \phi\left(x, y'\right)\right)}$$

- **למידה:** נרצה להביא למינימום את ה $log(MLE)$

$$L(w) = \log\left(\prod_{i=1}^{n} P\left(y_i \mid x_i; w\right)\right) = \sum_{i=1}^{n} \log P\left(y_i \mid x_i; w\right)$$

$$w^* = \arg\max_{w} L(w)$$

את הלמידה נעשה באמצעות שיטות מבוססות גרדיאנט. משום שהפונקציה קמורה.

$$\frac{\partial}{\partial w_j} L(w) = \sum_{i=1}^{n} \left(\phi_j\left(x_i, y_i\right) - \sum_{u'} P\left(y' \mid x_i; w\right) \phi_j\left(x_i, y'\right) \right)$$

עבור:

$$P(y \mid x; w) = \frac{e^{w \cdot \phi(x,y)}}{\sum_{y'} e^{w \cdot \phi(x,y')}}$$

- **החיזוי:** נמקסם את ההסתברות של התוית y בהינתן x.

$$argmax_y p(\mathrm{y} \mid \mathrm{x}; \mathrm{w}) = argmax_y score(\mathrm{y}, \mathrm{x}; \mathrm{w})$$

למעשה נכפול את הבלוק נוטיישן במשקולות w, ונבחר את הלייבל שהמשקל שלו הגבוה ביותר.

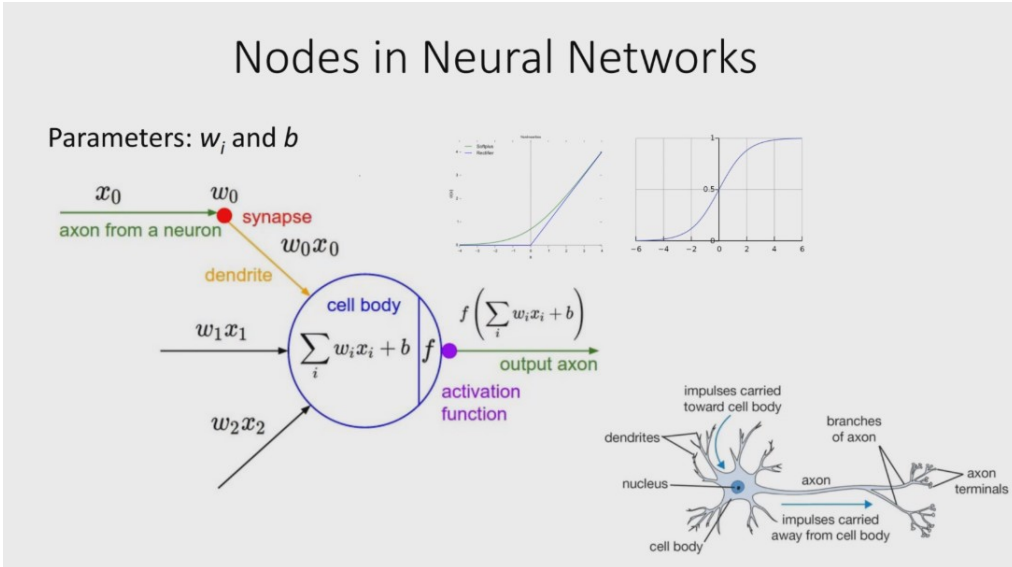
2.3.3 פרמטר רגולריזציה:

נרצה להוסיף פרמטר λ שיעזור לנו לבחור בצורה טובה יותר את התוית. עם פרמטר רגולריזציה הלמידה תראה כך:

$$L(w) = \sum_{i=1}^{n} \log p\left(\dot{y}_i \mid x_i; w\right) - \frac{\lambda}{2} ||w||^2$$

3 רשתות נוירונים:

- כל נוירון מהווה יחידה פשוטה שמחשבת חישוב פשוט, אך הרשת כולה תעשה חישובים מאד מורכבים - הרכבה של מלא פונקציות פשוטות שהופכות לבסוף לפונקציה מאד מורכבת.
- **הרעיון:** כל נוירון מקבל קלט מנוירון אחר, וכל קלט יוכפל במשקל w, שמסמן עד כמה האינפוט של הנוירון הזה חשוב. כל נוירון יסכום את כל האינפוטים עם המשקולות יוסיף פרמטר b, ויפעיל עליה פונקציית אקטיביציה.
- **פונקציית אקטיביציה:** פונקציה שמתנהגת מסויים עד נקודה מסויימת, ואחרי נקודה זו היא מתנהגת אחרת. המטרה היא להפוך את הקשרים ללא לינארים.
- **זה ייראה כך:**



- **פונקציית** Los**:** פונקציה שתגיד לנו כמה הרשת צודקת על הפרמטרים של הרשת - משקולות w והפרמטר b. בהינתן סט אימון.

$$L_S(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell\left(\theta; (x_i, y_i)\right)$$

- **האלגוריתם -** $backpropogation$**:** האלגוריתם איתו הרשת עובדת למעשה הוא מחשב לאחור את הגרדיאנט.

- **השכבות הנסתרות:** הן **אינן** פוקנציות קמורות, יהיו לנו מלא מינימומים מקומיים ולכן יהיה קשה מאוד למצוא מינימום גלובאלי. לכן נמצא את המינימום המקומי ונתכתס אליו, אך אין הבטחה שהא אופטימלי.

- $log\ linear$ **לעומת רשת נוירונים:** ניתן לחשוב על מודל $log\ linear$ כעל רשת נוירנים עם פלט בינארי. ניתן להניח כי ה$score$ של $class$ אחד הוא המינוס של ה $score$ של ה $class$ השני.

$$P(\text{ CLASS }_1 \mid x; w) = \frac{e^{w^T\phi(x,y)}}{e^{w^T\phi(x)} + e^{-w^T\phi(x)}} = \frac{1}{1+e^{-2\cdot w^T\phi(x)}}$$

באופן דומה ניתן לייצג בעזרת רשתות: האינפוטים יהיו הערכים של הפיצ'רים, כאשר האאוטפוט של נוירון תהיה פונקציית אקטיביציה - סיגמואיד, באופן הבא:

$$h_{w,b}(z) = f\left(w^\top z + b\right)$$
$$f(u) = \frac{1}{1+e^{-u}}$$

- $one-hot\ vectors$**:** האינפוט של הרשת הוא ווקטור ממשי, כשנרצה להכניס פיצ'רים קטגוריאלים נצטרך להפוך אותם לוורקטור ממשי.

נעשה זאת כך: לדוגמה עבור מילים - נקח ווקטור יחידה באורך אוצר המילים ונשים 1 בכל מקום שבו המילה מופיעה.

- **שכבת** $SOFTMAX$: כשנרצה ששכבה אחת תהיה ווקטור התפלגות ולא ווקטור ממשי, נוכל להפוך את הווקטור הממשי לקטור התפלגות באופן הבא:

$$SOFTMAX(\vec{x})_i = \frac{e^{x_i}}{\Sigma_i e^{x_i}}$$

4 $Sequence\ Methods$:

נמדוד הצלחה בעזרת $accuracy$: נחלק את המילים שצדקנו בהן חלקי כל המילים ב $corpus$.
מילה שנמצאת יותר פעמים ב $corpus$ יש לה השפעה גדולה יותר על ה $accuracy$.

4.1 בעיית $part\ of\ speach\ (POS)$:

- **הבעיה:** אנו מקבלים משפט כקלט, ואנו צריכים לשייך לכל מילה את חלק הדיבר - תואר, שם תואר, פועל וכו..
  נניח כי יש לנו למידה מפוקחת והמשפטים בקלט מתוייגים.

- **אבחנה ראשונה:** יש מילים מסוימות שהן שמות תואר ולכן הן יבואו במיקום מסויים של משפט מתבנית מסויימת, בנוסף ניתן להוסיף להן הטיות כדוגמת er או est.

- **אבחנה שניה - עמימות:** מילים שיש להן כפל משמעות, וצריך לתייג אותן לחלק הדיבר הנכון גם בקשר למשפט בו הן נאמרו. ולכן נצטרך לקבוע לכל **מופע** של מילה לחלק הדיבר אליו הוא שייך.

- **אילו פיצ'רים יעניינו אותנו:** לכל מילה יש אוסף מסויים של קטגוריות אליהן היא יכולה להשתייך. כדי שנוכל לסווג באופן מוצלח יותר לקטגוריה הרלוונטית למופע הנוכחי, נסתכל על המילים הנוספות במשפט וכך נדע את הקטגוריה הספציפית של המילה עליה אנו מסתכלים.
  1: נסתכל על ההסתברות של מילה להשתייכות לקטגוריה ($part\ of\ speach$).
  2: נסתכל על קטגוריות שנפוצות כשרשרת - אם שרשרת הקטגוריות לא הגיונית אזי הסיווג לא נכון, ולכן המילה שייכת לקטגוריה אחרת.
  3: אותיות גדולות וקטנות.
  4: תחיליות וסופיות - סיומת ed למשל יכולה להעיד על פועל.

4.1.1 אלגוריתם ראשון לפתרון הבעיה - $Sequence\ Labeling\ as\ Classification$:

- **הרעיון:** נתייג כל מילה באופן בלתי תלוי, אך נסתכל על המילים שמסביבה (מלפניה או אחריה) כדי שיעזרו לנו לתייג את המילה הנוכחית.
  יש שתי גישות - $Farward\ Backward$ האם נקח את המילים שלפני המילה או שאחריה.

- **החסרון:** יכול להיות שבהסתכלות על מילה אחת, המילים שמסביבה לא בהכרח יעידו על המילה הנוכחית, משום שהיא תלויה במילים רחוקות יותר.

בנוסף איך נדע אם לקחת את המילים שלפני המילה או מאחוריה, שהרי לעיתים זה משתנה ויש מילים שהמילה שמכריעה אותן תהיה לפניהן ויש מילים שהמילה שתכריע אותן תהיה אחריהן.

4.1.2 הפתרון - מודל לרקוב - $Hidden\ Markov\ Model\,(HMM)$:

- **הרעיון:** נסתכל על משפט x עם פרדיקציות מתאימות y. נרצה התפלגות משותפת על שרשראות של משפטים ותיוגים, ונמצא את השרשרת $y_1...y_n$ הסבירה ביותר עבור המשפט הנתון.
  נרצה להחזיר את התיוג ל y-ים שממקסמים לנו את ההסתברות.

- **ההנחה:** ההסתברות המשותפת של $x_1...x_n$ תתפרק למכפלת הגורמים הבאים כך (עבור מודל ביגראם) -

$$p\left(x_1 \ldots x_n, y_1 \ldots y_n\right) = q\left(STOP \mid y_n\right) \prod_{i=1}^{n} q\left(y_i \mid y_{i-1}\right) e\left(x_i \mid y_i\right)$$

כאשר q מסמן את ההסתברות לקבל את הקטגוריה ה i בהינתן הקטגוריה $i-1$. ו e מסמן את ההסתברות לקבל את הקטגוריה y_i בהינתן המילה x_i (ההסתברות שהמילה שייכת לקטגוריה).

- **הנחות על המודל:**
  הנחה ראשונה: המילה ה x_i אינה תלויה בשאר המשתנים בהינתן y_i.
  הנחה שניה: התגיות y מהוות שרשרת מרקוב והן ב"ת בתגיות הקודמות.

- **הערה:** כמו מודלי מרקוב לשפה (ביגראם, טריגראם) ניתן להגדיר מודל HMM מסדר גבוה יותר.

- **הבעיה עם ההנחות:** ישנם משפטים שלא מקיימים את הנחות וכן יש תלות בין כמה מילים ותוויות. **אך** אם נקח מודל מסדר גבוה יותר הוא יכסה את רוב המשפטים הקיימים.

- **מודל טריגראם:**

Trigram HMM

For any sentence $x_1 \ldots x_n$ where $x_i \in \mathcal{V}$ for $i = 1 \ldots n$, and any tag sequence $y_1 \ldots y_{n+1}$ where $y_i \in \mathcal{S}$ for $i = 1 \ldots n$, and $y_{n+1} = \text{STOP}$, the joint probability of the sentence and tag sequence is

$$p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i | y_i)$$

where we have assumed that $x_0 = x_{-1} = *$.

- **איך נלמד את הפרמטרים:** את הלמידה נעשה באמצעות MLE באופן הבא (עבור טריגרם):

$$\hat{q}\left(y_3 \mid y_2, y_1\right) = \frac{\#\left(y_1, y_2, y_3\right)}{\sum_y \#\left(y_1, y_2, y\right)}$$
$$\hat{e}\left(x_1 \mid y_1\right) = \frac{\#\left(x_1, y_1\right)}{\sum_x \#\left(x, y_1\right)}$$

כאשר $\#\left(y_1, y_2, y_3\right)$ מסמן $count$ - מספר הפעמים שהופיעה הסדרה $\left(y_1, y_2, y_3\right)$

שיטות החלקה - $Smoothing$:

- **שיטות החלקה** $smoothing$**:** ניתן להשתמש גם פה בשיטות החלקה, כמו במודלי שפה. למשל להגדיר λ לכל מודל ולחשב כמה מודלים במקביל.

- **פסואודו מילים -** $Smoothing\ with\ Pseudowords$**:** נגדיר פסואודו מילים - מילים שלא מופיעות הרבה בטקסט ולכן אנו נתייג אותן באופן עצמאי לקטגוריה שנבחר (לדוגמה ספרות המסמלות שנה, או קוד מוצר). ונחליף את כל המילים באותה הקטגוריה הנדירה - בשם הקטגוריה. באופן זה הפסואודו־וורד כן יופיע כקטגוריה.

הסקה - $Inference$**:**

- **מה נרצה:** נרצה למצוא את סדרת התיוגים y בהינתן משפט x. נשים לב כי לכל x_i יש y_n קטגוריות אפשריות. ולכן עבור משפט באורך n יהיו לנו $\left(y_n\right)^n$ אפשרויות.

- **הפתרון - אלגוריתם** $Viterbi$**:** אלגוריתם דינמי. נבנה גרף שכבות מכוון, כך שבכל שכבה יש קודקודים כך שכך קודקוד מייצג קטגוריה. מלבד השכבה הראשונה שתכיל קדקוד בודד שייצג את $START$, והשכבה האחרונה תכיל קודקוד בודד שייצג את $STOP$. נמתח צלעות בין קדקודים מכל שכבה לכל שכבה הבאה, כך שכל מסלול ייצג השמה, וכל מסלול יקבל משקל ששווה להסתברות (כל קשת תקבל את משקל ההסברות להגיע דרכה לקודקוד הבא, בהינתן הקודקוד הקודם).
לאחר מכן נחפש את המסלול עם המשקל הגבוה ביותר.
האלגוריתם: עם מציאת סדרת ה y־ים, נזכור את המסלול ומהיכן הגענו עבור כל קודקוד.

The Viterbi Algorithm with Backpointers

Input: a sentence $x_1 \ldots x_n$, parameters $q(s|u,v)$ and $e(x|s)$.

Initialization: Set $\pi(0,*,*) = 1$

Definition: $\mathcal{S}_{-1} = \mathcal{S}_0 = \{*\}$, $\mathcal{S}_k = \mathcal{S}$ for $k \in \{1 \ldots n\}$
Algorithm:

- For $k = 1 \ldots n$,
  - For $u \in \mathcal{S}_{k-1}$, $v \in \mathcal{S}_k$,

$$\pi(k,u,v) = \max_{w \in \mathcal{S}_{k-2}} \left(\pi(k-1,w,u) \times q(v|w,u) \times e(x_k|v)\right)$$
$$bp(k,u,v) = \arg\max_{w \in \mathcal{S}_{k-2}} \left(\pi(k-1,w,u) \times q(v|w,u) \times e(x_k|v)\right)$$

- Set $(y_{n-1}, y_n) = \arg\max_{(u,v)} \left(\pi(n,u,v) \times q(\mathsf{STOP}|u,v)\right)$
- For $k = (n-2) \ldots 1$, $y_k = bp(k+2, y_{k+1}, y_{k+2})$
- **Return** the tag sequence $y_1 \ldots y_n$

6

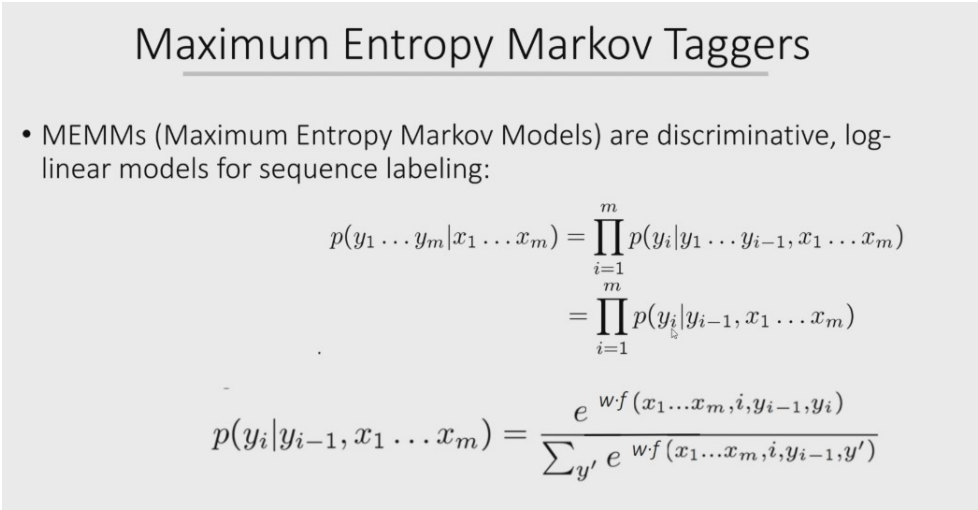
- **זמן ריצה של האלגוריתם:** יהיו לנו n שכבות כמספר המילים במשפט, ועבור כל שכבה יהיו לנו k קודקודים כמספר הקטגוריות.
  עבור מודל מסדר l זמן הריצה הינו $O\left(n \cdot |S|^l\right)$. כלומר קבוע ב n.

4.1.3 מודלים דסקרמינטיבים:

הגדרה 4.1 מודל דסקטמינטיבי הוא מודל שמגדיר את ההתפלגות המותנה של ה y־ים (קטגוריות) בהינתן ה x־ים (מילים).

אלגוריתם ראשון ־ $Maximum\ Entripy\ Markov\ Taggers\,(MEMM)$**:**

- **עבור כל משפט נחשב את ההסתברות הבאה:**



- **ההנחות של האלגוריתם:**

1: בהינתן ה x־ ים ה y־ים מהווים שרשרת מרקוב (הנחה דומה ל HMM). הנחה זו מאפשרת לנו להתנות רק על y_{i-1} כי יש אי תלות לכל ה y־ ים שלפני y_{i-1}.

2: הצורה של כל גורם במכפלה היא $log\ linear\ model$ ולכן היא שווה לאקספוננט חלקי סכום האקספוננטים (מה שמופיע בשקף).

- **תזכורת ־** $log\ linear\ model$**:** לכל מכפלה יש וקטור פיצ'רים ־ f, ווקטור משקולות w שמגדיר מה החשיבות של כל פיצ'ר, ואנו מכפילים את ווקטור המשקולות בווקטור הפיצ'רים.
  נרצה ללמוד: את ווקטור המשקולות w.

- **איך נגדיר את ווקטור הפיצ'רים** f**:** נגדיר ווקטור שמכיל פיצ'רים עבר קומבינציות בין מילים לקטגוריות כפי שנקבע. לדוגמה ־ נוכל לגדיר כי מילה שנגמר עם "ing" ומופיעה עם תגין מתאימה לפעלים ־ V תקבל משקל גדול באותו הפיצ'ר.

- **למידת המודל:** נלמד באמצעות MLE ־ מקסום הנראות.

MLE in MEMMs

- Recall the model: $$Pr(y|x) = \prod_{i=1}^{N}\prod_{j=1}^{n(i)} Pr(y_j^{(i)}|y_{j-1}^{(i)}, x^{(i)}) = \prod_{i=1}^{N}\prod_{j=1}^{n(i)} \frac{exp(f(j, y_j^{(i)}, y_{j-1}^{(i)}, x^{(i)})^T \cdot w)}{Z(y_{j-1}^{(i)}; w)}$$
- The partition function: (locally normalized)
$$Z(y_{j-1}^{(i)}; w) = \sum_{y_j} exp(f(j, y_j, y_{j-1}^{(i)}, x^{(i)})^T \cdot w)$$
- Log-likelihood (where $(x^{(i)}, y^{(i)})$ is the training data):
$$LL(w) = \sum_{i=1}^{N}\sum_{j=1}^{n(i)} \left[f(j, y_j^{(i)}, y_{j-1}^{(i)}, x^{(i)})^T \cdot w - log(Z(y_{j-1}^{(i)}))\right]$$
- Gradient for w: (T is the set of labels)
$$\frac{\partial LL}{\partial w_k} = \sum_{i=1}^{N}\sum_{j=1}^{n(i)} \left[f_k(j, y_j^{(i)}, y_{j-1}^{(i)}, x^{(i)}) - \sum_{y' \in T} Pr(y_j = y'|y_{j-1}^{(i)}, x^{(i)}) \cdot f_k(j, y_j = y', y_{j-1}^{(i)}, x^{(i)})\right]$$

הסקה - $Inference$:

נרצה למצוא את סדרת ה y־ים הסבירה ביותר עבור ה x ־ים . נשתמש בתכנון דיני כאשר בכל שלב נחפש את המקסימום עבור תחילית מסויימת של סדרת תגיות, (דומה לאלגוריתם של $Viterby$).

- **האלגוריתם:**

Inference in MEMMs

- Define n to be the length of the sentence
- Define
$$r(t_1 \dots t_k) = \prod_{i=1}^{k} q(t_i|t_{i-2}, t_{i-1}, x_{[1:n]}, i)$$
- Define a dynamic programming table

$\pi(k, u, v)$ = maximum probability of a tag sequence ending in tags u, v at position k

that is,
$$\pi(k, u, v) = \max_{\langle t_1, \dots, t_{k-2}\rangle} r(t_1 \dots t_{k-2}, u, v)$$

כאשר - r הוא מכפלה של k הגורמים הראשונים, עבור אינדקס רץ k. אנו ננסה למקסם את מכפלת הסדרות החלקיות. ונחזיק בטבלה הדינמית את המקסימום הסתברות עבור מכפלה של k גורמים שנגמרת ב (u, v)

נוסחת הבסיס וההתקדמות:

Base case:

$$\pi(0, *, *) = 1$$

Recursive definition:
For any $k \in \{1 \dots n\}$, for any $u \in \mathcal{S}_{k-1}$ and $v \in \mathcal{S}_k$:

$$\pi(k, u, v) = \max_{t \in \mathcal{S}_{k-2}} \left(\pi(k-1, t, u) \times q(v|t, u, x_{[1:n]}, k)\right)$$

where $\mathcal{S}_k$ is the set of possible tags at position k

- **בעיות עם המודל:** הגדרנו כי סכום כל הקשתות שיוצאות מקודקוד מסויים שווה ל 1, כי זאת הסתברות וחילקנו בגורם הנרמול Z.

הבעיה: לעיתים עבור קודקוד שההסתברות להגיע אליו היא נמוכה, לאחר שהגענו אליו - ההסתברות הגבוהה ביותר לעבור ממנו הלאה תהיה אם הקודקוד יחזור לעצמו, לכן אנו לרוב לא נתקדם אלא נחזור לאותו הקודקוד (אותה התגית) שוב ושוב.

הפתרון: כל צלע תהיה בעלת מספר אחר, ומשקל כל הצלעות שיוצאות מקודקוד מסיים לא בהכרח ייסכם ל 1, אלא למספר חיובי כלשהו. **אך** סכום כל המסלולים ייסכם ל 1 (למעשה זה מה שרצינו מלכתחילה ולכן הגבלנו את סכום הקשתות להיות 1).

המודל החדש - $Globally\ normalized\ Models$ מודל CRF:

במודל זה סכום **המסלולים** יהיה 1, כלומר - במקום גורם נרמול לוקאלי על פני המעברים, יהיה לנו גורם נרמול גלובאלי על פני כל המסלולים.

- **הנרמול החדש ייראה כך:** נחלק בסכום כל המסלולים האפשריים.

Sequence Conditional Random Fields (CRFs)

- CRFs are similar to MEMMs, but the normalization is global.
- This solves the label bias: the scores of the next state given the current one need not sum up to 1
- That is, given sequences $(x^{(1)}, x^{(2)}, \dots, x^{(N)})$ and labels $(y^{(1)}, y^{(2)}, \dots, y^{(N)})$, we define:

$$Pr(y|x) = \prod_{i=1}^{N} Pr(y^{(i)}|x^{(i)}) = \prod_{i=1}^{N} \frac{\prod_{j=1}^{n(i)} exp(f(j, y_j^{(i)}, y_{j-1}^{(i)}, x^{(i)})^T \cdot w)}{Z(x^{(i)}; w)}$$

$$Z(x^{(i)}; w) = \sum_{y} \prod_{j=1}^{n(i)} exp(f(j, y_j, y_{j-1}, x^{(i)})^T \cdot w)$$

- **למה הבעיה נפתרה:** באלגוריתם הקודם מכיוון שכל הקשתות נסכמו ל 1 היינו חייבים לתת לצלע אחת משקל קרוב מאד ל 1. לכן לאורך זמן נתקענו באותו המסלול, אך באלגוריתם הנוכחי אנו נאפשר צלעות עם משקלים שלא נסכמים ל 1, ונוכל לתת לצלע הממשיכה משקל מאד קטן. לכן לא ניתקע באותו קודקוד כי לא נבחר אותה.

- **איך נלמד:** נמצא את w עם MLE בעזרת הגרדיאנט באופן הבא - ($LL = log\ Likelihood$)

Estimating w

- We use maximum likelihood estimation (sometimes with l_2 regularization):

$$LL(w) = \sum_{i=1}^{N} \Big[\sum_{j=1}^{n(i)} f(j, y_j^{(i)}, y_{j-1}^{(i)}, x^{(i)})^T \cdot w - log(Z(x^{(i)}; w)) \Big]$$

- No close formula. We use gradient-based methods:

$$\frac{\partial LL}{\partial w_k} = \sum_{i=1}^{N} \Big[\sum_{j=1}^{n(i)} f_k(j, y_j^{(i)}, y_{j-1}^{(i)}, x^{(i)}) - \boxed{\sum_{y \in T^{n(i)}} Pr(y|x^{(i)}) \cdot \sum_{j=1}^{n(i)} f_k(j, y_j, y_{j-1}, x^{(i)})} \Big]$$

- **הקושי בחישוב החלק שנמצא במבן הכחול:** חישוב משקל כל המסלולים גרף הוא אקספוננציאלי, משום שיש מספר אקספוננציאלי של מסלולים. לכן זה מקשה על חישוב הגרדיאנט.

- **נפשט את החלק במלבן:** במקום להסתכל על כל המסלולים בגרף, נסתכל על כל קשת ונכפול אותה בסכום כל משקלי המסלולים שעוברים דרכה, ונכפול בפיצ'רים f_k.
כך ירדנו מחישוב אקספוננציאלי לפולינומי (מספר הצלעות). וכעת הקושי הוא לחשב את סכום כל משקלי המסלולים שעוברים דרך קשת מסויימת.

$$\sum_{j=1}^{n(i)} \sum_{y_j, y_{j-1}} Pr\left(y_j, y_{j-1} \mid x^{(i)}\right) \cdot f_k\left(j, y_j, y_{j-1}, x^{(i)}\right)$$

- **כיצד נחשב את סכום כל משקלי המסלולים שעוברים דרך קשת מסויימת -** $Pr\left(y_j, y_{j-1} \mid x^{(i)}\right)$**:** ניתן לפתור את זה עם תכנון דינמי כך - נחלק את החישוב לני חלקים.
α**:** עבור כל קודקוד v נחשב את סכום המשקלים של כל המסלולים שמגיעים עד לקודקוד v (מספר אקספוננציאלי).
β**:** עבור כל קודקוד v נחשב את סכום המשקלים של כל המסלולים שמגיעים עד לקודקוד הסופי y_n ומחילות בקודקוד v (מספר אקספוננציאלי).
אנו יכולים לחסוך את החישוב האקספוננציאלי בכל שב בעזרת חישוב דינמי, כך שעבור כל קודקוד אנו לוקחים את כל קודקודי השכבה שלפניו ומכפילים בצלעות שיוצאות להם ומגיעות אליו, וסוכמים.
האלגוריתם - $Forward\ Backward$**:**

CRFs: Forward-Backward Algorithm

- The parameters can be computed using dynamic programming with these formulas (for *i*=1, computing α, same for β with *i*=n)

$$\alpha_i(y) = \sum_{y'} M_i(y', y)\alpha_{i-1}(y') \qquad \beta_i(y) = \sum_{y'} M_{i+1}(y, y')\beta_{i+1}(y')$$

- The marginal can now be computed as:

$$Pr(y_j, y_{j-1}|x^{(i)}) = \frac{M_j(y_j, y_{j-1}) \cdot \beta_j(y_j) \cdot \alpha_j(y_{j-1})}{\boxed{Z(x)}}$$

- **כיצד נמצא את סכום כל המסלולים שעברו דרך צלע** $e = (u, v)$**:** נסתכל בטבלה שלנו ונקח את סכום כל המסלולים שהגיעו מ y_0 עד u **כפול** כל המסלולים מ v על y_n, **ונכפול** במשקל הצלע e.

- **כעת נרצה לחשב את** $Z(x)$**:** למעשה הוא סכמיה על פני כל המסלולים בגרף (נוכל לחשב עם כל ה α של השכבה האורונה, או באופן שקול עם כל ה β של השכבה הראשונה).

הסקה - $Inference$**:**
נוכל לעשות עם $Viterby$ באותו האופן של אלגוריתמים קודמים. כי החלוקה ב Z לא משנה לנו, כי אנו מנסים למצוא את ה y הסביר ביותר שממקסם את המונה בהינתן סדרת x־ים . ונוכל למקסם אותו עם מכפלה של משקלי הצלעות.

4.2 בעיית $Named\ entity\ recognition\ (NER)$:

- **הבעיה:** אנחנו מקבלים מסמך עם משפטים ואנו צריכים לקבוע מהם השמות - $Named\ entity$ בטקסט. בנוסף לקבוע עבור כל אחד מהם מה הוא - מדינה, ארגון אדם וכו. ניתן לחשוב על בעיה זו כעל תיוג מילים לקטגוריות אנו מקבלם דאטה מתוייג - למידה מפוקחת. מילה שלא שייכת לאף קטגוריה תתוייג כ NA.

5 $Word\ Embeddings$:

5.1 שיטות $distributional\ Semantic$:

מה נרצה: אנו רוצים לייצג מילים באמצעות ווקטור, כך נוכל ללמוד על מילים דומות בעזרת הכיוון אליו ווקטורים שמייצגים מילים מצביעים. וללמוד הקשרים של מילים.
הנחת $distributional\ hypothesis$**:** מילים שנוטות להופיע בהקשרים דומים, דומות אחת לשניה מבחינת המשמעות, ולהיפך. כלומר יש הקשר סמנטי בין מילים דומות.

5.1.1 שיטות מבוססות ספירה - $count - base\ Models$:

- **הרעיון:** נבחר חלון של המילה w וניצור ווקטור עבור המילה w, כאשר הווקטור מכיל את כל המילים שהיו בחלון של המילה w, עם מספר הפעמים שכל מילה הופיעה בכל חלון.
ההנחה אומרת כי מילים שיהיו באותה הקטגוריה עם המילה w, יהיה להן ווקטור שדומה לווקטור של המילה w.

- **כיצד נבדוק דמיון בין מילים בעזרת הווטורים שלהן:** נחשב את הדמיון בעזרת cos הזווית בין הווקטורים:

$$similarity(w,u) = \frac{w \cdot u}{\|w\|\|u\|} = \frac{\sum_{i=1}^{n} w_i u_i}{\sqrt{\sum_{i=1}^{n} w_i^2}\sqrt{\sum_{i=1}^{n} u_i^2}}$$

אם הווקטורים מצביעים לאותו הכיוון (יש דמיון) נקבל 1 , אחרת אם הם מנוגדים נקבל 1- , אם הם חופפים חלקית נקבל מספר באמצע הטווח.

- **הבעיות:**
1: הווקטורים שנקבל יהיו נורא דלילים ($sparsity$), כי מילים רבות יקבלו את הערך 0 בווקטור.
2: תהיה לנו יתירות של מילים דומות, לדוגמה המילים רכב ואוטו יופיעו בווקטור כשתי מילים שונות למרות שהן דומות. כך אם מילה אחת מופיעה הרבה פעמים עם אוטו אך לא עם רכב, ומילה אחרת תופיע הרבה פעמים עם המילה רכב אך לא עם אוטו, המודל יסיק כי הן שונות. למרות שהן מאותה הקטגוריה.
3: לא יהיה לנו ייצוג מספיק למילים נדירות.

- **הפתרון:** ניתן להוריד מימד כדי לפתור את הבעיות של מילים דומות שמופיעות בנפרד בווקטור.
נוכל לעשות זאת ע"י מעבר על הווקטור ובדיקת הדמיון בין המילים בווקטור, ולצמצם אותן.

5.1.2 שיטות חיזוי - $Prediction - based\ Models$:

- **הרעיון:** במקום לייצג את השכנים של המילה בתור ווקטור ארוך. נייצג כל מילה באמצעות וקטור ממשי, הווקטור ייצג את התפלגות השכנים של המילה, כך שנוכל לחלץ את התפלגות השכנים של המילה מהווקטור.

- **תובנות:**
1: אנו לא נוכל לעקוב אחרי הווקטור כי אין לנו באמת מושג מה הוא מייצג.
2: אנו נשתמש בכלים של למידה מפוקחת - רשת נוירונים.

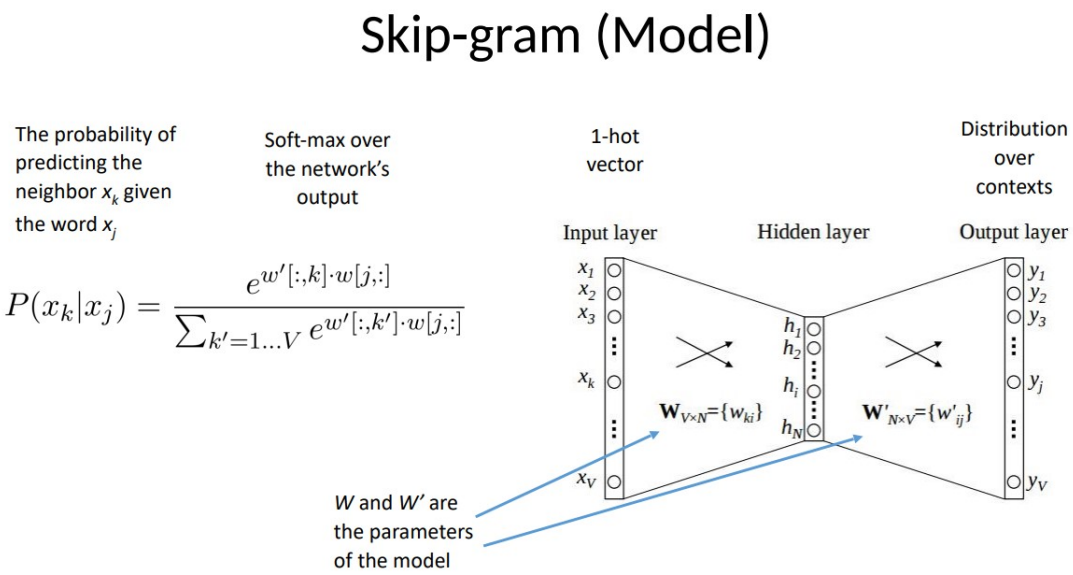
- **נהפוך את הבעיה לבעיה מפוקחת כך:** בהינתן מילה בתוך משפט אנו ננסה לחזות מיהן המילים השכנות שלה. כך נחשוב על הבעיה מהצד השני - ננסה לחזות את ווקטור התפלגות השכנים של המילה, בהינתן המילה.
נשים לב כי רוב הפעמים אנו נטעה, כי מן הסתם ננחש מילים שכנות לא נכונות.
היתרון: נוכל להצליח אם נדע את ההתפלגות של המילים, אנו לא צריכים לדעת כלום חוץ מהתפלגות של המילים השכנות בהינתן המילה w.

מודל $Skip - gram$:

- **מודל** $Skip-gram$:

נוטציות: את המילה ה j במילון נסמן ב x_j. את גודל אוצר המילים נסמן ב V (מספר המילים x_j השונות). ו N יהיה היפר פרמטר שקובע את המימד.

נבנה רשת באופן הבא:



- **איך נגדיר את הרשת:**

1: עבור כל מילה נגדיר ווקטור מילים כך שהמיקום בווקטור שמייצג את המילה מסומן ב 1, וכל שאר המקומות באפסים.

2: לאחר מכן נכפיל את הווקטור במטריצה W מגודל $V \times N$ (שקול ללקחת שורה אחת מהמטריצה כי הווקטור מכילאפסים בכל הקאורדינטות למעט אחת).

3: את הווקטור שקיבלנו נכפיל במטריצה W' עם מימדים הפוכים $N \times V$, כך שכל עמודה במטריצה מתאימה למילת הקשר, ולאחר המכפלה של הווקטור במטריצה נקבל וקטור y שמכיל $score$ עבור כל מילה.

4: לאחר מכן נפעיל על התוצאה $softmax$ - העלאה באקספוננט ונירמול, כדי ליצור ווקטור הסתברות.

- **הפרמטרים שנרצה ללמוד:** הן המטריצות W, W'.

- **שלב האימון:** אנו מנסים למצוא את המקסימום הבא (למקסם את הסתברות זוגות המילים שהופיעו בסט האימון) -

$$argmax_{W,W'} \log[P(\text{ text })] =$$
$$argmax_{W,W'} \sum_{(x_k,x_j)\in \text{ text}} \log\left[P\left(x_k \mid x_j\right)\right]$$

עבור X_k, X_j הן כל זוג מילים שלכל אחת מהן יש לכל היותר k טוקנים.

- **איך נחלץ את ה** $Embedding$: נסתכל על השכבה הנסתרת (W, W') כעל סוג של $Embedding$. נשים לב שיש לנו שני סוגים - המילה עליה אנו מסתכלים (אימדניג הקלט), והאימבדינג של מילות ההקשר (אימבדינג הפלט).

כדי לדעת אם שתי מילים w, w' קשורות, נקח מהמטריצה W את השורה שמתאימה למילה w, ונכפול (מכפלה

סקלרית) בעמודה מ W' שמתאימה למילה w' וזה יהיה ה $score$. לאחר מכן נפעיל סופטמקס על התוצאה. ככל שהתוצאה גבוהה יותר, אזי הקשר בין המילים חזק יותר.

6 $Sentiment\ Analisis$:

- **המשימה:** אנו רוצים להסיק מהטקסט מה הכותב ניסה לומר - מה העמדה שלו בעניין. לדוגמה עבור ביקורת על סרט האם היא הייתה חיובית או שלילית.
  דוגמה נוספת היא חיזוי מציוצים בטוויטר האם מניה הולכת לשנות את ערכה בעקבות באז שנוצר סביבה.
  הערה: אנו נתמקד בביקורת לאורך זמן - $Attitudes$, ולא בביקורת אימפולסיבית כתוצאה מבעיה רגעית.

- **נרצה להפריד בין סנטימנטים:** מי כתב את המקור, על מה מדובר, מה סוג הסנטימנט - חיובי שלילי, מה סוג הטקסט - משפט, מילה, מסמך שלם.

- **במה נתמקד:** אנו נתמקד במשפטים שלמים ולא מורכבים, ותיוג ביקורות.

- **מסווגים פשוטים לפרון הבעיה:** ניתן להשתמש במודלי $n-gram$, מודלים לוג לינארים ועוד שיפתרו את הבעיה בצורה טובה.

- **הבעיות של המודלים הפשוטים:**
  1 סרקזם: מודלים אלו ייתקשו להתמודד עם סרקזם. ביקורת שלילית שנכתבת בצורה חיובית.
  2: ביקורת שלילית שנכתבת בקצרה בסוף הטקסט - $Thwarted\ Expectations\ and\ Ordering\ Effects$: יהיו יותר מילים חיוביות אך הביקורת שלילית.

- **שיטה אחת:** מרגע שנראה מילה שלילית, נוסיף לכל מילה שבאה אחריה את התגית Not , עד שנגיע לסימן פיסוק. כך לא נספור את המילים שבאות אחריה כחיוביות.

6.1 $RNN - Recurrent\ Neural\ Networks$:

- **ייצוג המידע:** עבור כל $state$ יהיה לנו ווקטור, ופונקציית מעבר שאותה נרצה ללמוד, הפונקציה מעבירה אותנו בין $states$ שונים. הפונקציה מקבל את ה $state$הנוכחי ואת x - ומחזירה לנו את ה $state$ הבא. אנו נציג לרשת את התוצאות האם היא חיובית או שלילית עבור טקסט מסויים, והיא תצטרך ללמוד את הווקטורים לכל מילה.
  כך במקום להחזיק $states$ שמסתכלים על k מילים אחרונות, אנו נייצג את ההסטוריה עד המצב ה k, בעזרת קידוד של המידע החשוב מההסטוריה עד השלב הנוכחי.

- **למידה:** נתרגם את ווקטור המילים שלנן $x_1...x_n$ לסדרת $states$ ווקטורים $s_1...s_n$. את המיפוי נעשה בעזרת פונקצייה $Recurrent$ (רקורנטית) R שתביא לנו את ה $state$ הבא בהינתן הווקטור ה$state$ הקודם:

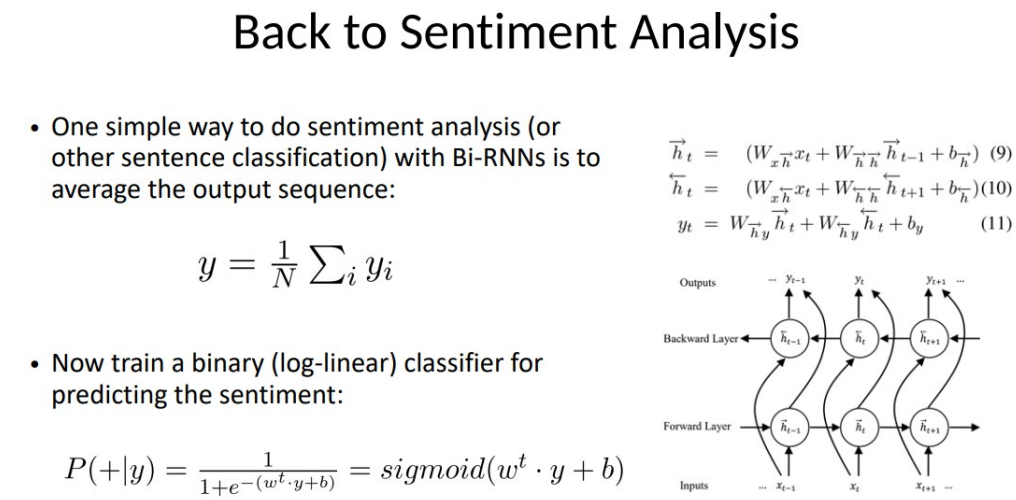
$$s_i = R\left(s_{i-1}, x_i\right)$$

אנו נרצה ללמוד את R, ואנו נשתמש באותה הפונקציה עבור כל הצעדים.
בנוסף תהיה לנו פונקציית אאוטפוט O שממפה בין ה $state$ לבין האאוטפוטים שאנו רוצים. אנו נרצה ללמוד את

הפונקציה הזאת גם כן.
כל קודקוד ברשת יפעיל את שתי הפונקציות R, O. והמשקולות w יהיו אותן משקולות לכל הקודקודים.

- **רשת** $Bi-RNN$**:** רשת דו כיוונית, כך שיש לנו שתי שכבות, אחת הולכת קדימה והשניה הולכת אחורה. בדרך זו נוכל למצוא את כל המילים שרלוונטיות למילה הנוכחית, בין אם הן נמצאות מאחוריה או מלפניה.
שכבת האאוטפוט תכריע איזו שכבה חשובה יותר בחיזוי.

- **כיצד נשתמש:** נגדיר אאוטפוט של כל מילה - y, אך הוא לא יהיה הסנטימנט של המילה עצמה, אלא ווקטור נוסף. לאחר שנמצא את הווקטורים של המילים בעזרת הרשת, נאמן עליהם מודל לוג לינארי.



- **למידה בעזרת פולקציית** $Loss$**:** עבור כל טקסט נחזה עד המילה ה i, מה ההסתברות לקבל את המילה הבאה. וננסה למקסם את ההסתברות על המילה הבאה בהינתן i המילים הקודמות.

- **איזו פונקציית** $Loss$ **נבחר:** נקח את log ההסתברות על המילה הנכונה.

- **כיצד נגדיל את אוצר המילים:** נוכל להסתכל על מספר מילים כעל מילה אחת - לדוגמה $new\ york$, או לנסות לחזות אותיות במקום מילים.
הטריידאוף הוא: אם ננסה לחזות אותיות במקום מילים, המודל יטעה לחזות מילים שהן לא מילים, ולכן נצטרך ללמד אותו איך לאיית ואילו מילים הן נכונות. בנוסף הוא ייצטרך ללמוד ממילים שמכילות אותיות דומות אך עם הטיות שונות. לכן נוכל ללמוד טוב יותר, אך נצטרך ללמוד הרבה יותר.

7 דקדוק ותחביר:

- **פסוקית:** חלק משפט שעומד בפני עצמו, לדוגמה - "רון פתח את הדלת". פסוקית מוגדרת לפי פעלים.

- **פסוקית מושא -** $complement\ clauses$**:** פסוקית מתוך משפט המכיל כמה פסוקיות, כך שהפסוקית הזאת היא מושא המשפט. לדוגמה - "יובל ראתה שאלי בעט בכדור", פסוקית המושא היא "אלי בעט בכדור".
לרוב אלו פסוקיות שיענו על שאלת "מה"?.

- **קורדינציה** - $Coordination$: מקרים שבהם נקח כמה פסוקיות ונהפוך אותן לרשימה. לדוגמה - "חזרתי הביתה התקלחתי, שתיתי ואכלתי".
בד"כ נשתמש במשפט ב and, or.

- **פסוקית לואי** - $Relative\ clauses$: פסוקית מקוננת בפסוקית אחרת, והיא הרחבה לאחד משמות העצם במשפט. לדוגמה - "כלב נובח לא נושך", פסוקית הלואי היא "לא נושך"

- $Linked\ clauses$: סוגים של חיבורים בין פסוקיות -$after, although, if$.

- **ביטויים שמניים** - $Noun\ Phrases$: יחידות בסיסיות שיכולות לשמש כארגומנטים במשפט - שמות פרטיים, שמות קטגוריות, מיידעים - The.

- **גרעין** - $Syntactic\ Head$: מגדיר את האובייקט עליו אנו מדברים. לדוגמה - חברת משלוחים" הגרעין הוא "חברה".
ניתן לייחס אותו גם לפסוקיות - **הפועל** הראשי בפסוקית הוא גרעין הפסוקית.

7.1 מבנים תחביריים:

- **הרעיון:** גישה לניתוח תחבירי בה אנו נקבל משפט ונבנה ממנו עץ, ע"י כך שלכל מילה נגדירה מה ה $head$ שלה, למעט השורש שלו אין $head$.

מבנים תחביריים:

- **למה להשתמש במבנה תחבירי:** מבנה תחבירי עוזר לנו לפרק את המשפט לתתי מחרוזות בעלי משמעות.

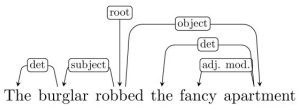
1: כשנרצה לתרגם מאנגלית לשפה שמשנה את מבנה המשפט, נוכל להחליף את מבנה המשפט לפי העץ התחבירי שלו.

2: המבנה התחבירי יכול לתת לנו ידע סמנטי על הטקסט, לרוב הנושא הוא המבצע, והנשוא הוא זה שמבוצעת עליו הפעולה.

3: ניתן להפיג עמימות ע"י המבנה התחבירי של המשפט. משפט שיש לו שתי משמעויות, יכול להיות מפורש ע"י המבנה התחבירי של המשפט. לדוגמה - "מחזיק כוסות מפלסטיק", לא ברור אם המחזיק או הכוסות מפלסטיק. והעץ התחבירי יכול לעזורלנו לפרש זאת

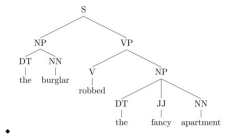
7.1.1 עץ תלויות:

- **עץ תלויות** - $Dependency\ Parses$: נהפוך את המבנה התחבירי של המשפט לגרף מכוון. הקודקודים הם המילים, ולכל מילה יש חץ שנכנס אליה ומגיע מהגרעין של המשפט.
נעשה זאת באופן רקורסיבי עבור כל יחידה. תחילה נמצא את הגרעין של המשפט כולו, ולאחר מכן נמצא את הגרעין של כל יחידה.



7.1.2 $Phrase-based(constituency)parses$:

- $Phrase-based(constituency)parses$: בייצוג זה המילים אינן קודקודים, אלא עלים של העץ.



כיצד נחבר את הצלעות בין מילים:

- **זהות המילים** - $Bi-lexical\ affinities$: נסתכל על זהות המילים, והיא תספר לנו אם יש צלע, ואם כן מה הטיפוס של הצלע.
  החסרון: יש ספרסיות, בנוסף כדי להשתמש בגישה זו אנו צטרך המון דאטה, כי מספר זוגות המילים שווה למספר המילים בריבוע.

- **אורך הצלע** $Dependency\ distance$: רוב הקשתות הן קצרות, לכן נסתכל על אורך הצלע, ככל שהצלע ארוכה יותר ההסתברות לקבל אותה יורדת.

- **כיוון הצלע** $Intervening\ mareria$l: כיוון הצלע מצביע על נכונות, כי לדוגמה בעברית פועל בא אחרי שם עצם.

- $Valency$: מילים מסויימות יכולות לקבל פעלים מסויימים בלבד. לדוגמה המילים לישון, למות לא מקבלות מושא. לעומת לתת ולהעניק יש להן שני מושאים בד"כ. לכן ניתן למצוא את הקשר בין מילים לפי תכונת הפועל.

מה נרצה להשיג:

- נרצה למקסם את מספר הצלעות שנמצאות ברפרנס - $gold\ standart$ והתיוג שלנו. נחשב כך:

$$attachment\ score = acc = \frac{correct\ edges}{num\ of\ words}$$

- $label\ VS\ unlabel$: שתי אפשרויות למדוד את ההצלחה, $unlabel$ ידרוש שרק הצלעות יהיו נכונות. לעומת זאת $label$ ידרוש כי גם בצלעות וגם התוית יהיו שוות.

7.2 $Graph\ Base\ Parsing$:

מודלי $MST\ Parser$:

- הפרידיקציה שנקבל בסוף תהיה מבנה מסויים. אנו נדבר על עץ פורש מקסימלי MST.

- **למידה ופרדיקציה:** נבנה מודל שיתן משקלים לצלעות, גרף שהמילים הם קודקודים עם צלעות ממושקלות בניהם. לאחר שנלמד אותו, בהינתן משפט אנו נבנה את כל הצלעות שלו ונפעיל עליו MST למצוא את העץ המקסימלי.

7.2.1 מודלים קונסיסטנטים:

- **הגדרת המודל:**

MST Parser

Define a scoring function over all possible directed trees over $V = \{w_1, ..., w_n, ROOT\}$ where $ROOT$ is the root of the tree. Let $\Phi : V^2 \times L \times S \to \{0,1\}^d$, where L is the label set and S is the set of sentences (feature values can also be real numbers if needed), be a feature function over possible edges.

Let θ be the weight vector (the parameters of the model):

$$score_\theta(v_1, v_2, l \,|x_{1:n}) = \theta^t \cdot \Phi(v_1, v_2, l, x_{1:n})$$

For a directed tree T define:

$$score_\theta(T \,|x_{1:n}) = \sum_{(v_1,v_2,l)\in T} score_\theta(v_1, v_2, l \,|x_{1:n})$$

- **נשים לב כי משקל העץ הוא סכום משקלי הצלעות.**

יתרון: כך נוכל למשקל כל אחד מהפיצ'רים, נקבל גרף מכוון וממושקל, ועליו נריץ אלגוריתם לפתרון הבעיה.

חיסרון: לא נוכל להתמודד עם צלעות אחרות שמשפיעות על הצלע שלנו. לדוגמה עבור משפט המכיל את המילים $Jhon, saw, Mary$ היינו ממדלים את שני שמות העצם עם אותו משקל להיות קשורים לפועל saw. למרות שברור כי שניהם לא יכולים להיות קשורים לאותו הפועל. לכן נרצה שאם בחרנו קשת אחת, נוכל לפסול קשת אחרת.

- **גישה נוספת באמצעות מודל לוג לינארי - הסתברות, גרדיאנט ופונקציית** $Loss$**:** לא נשתמש במודל זה כי הוא מסובך.

MST Parser: Inference and Learning

- Note that inference is simply finding the maximum directed spanning tree
  - We can score each edge based on its features and
  - This is done by the Chu-Liu Edmonds algorithm (not necessarily projective)
- It is possible to define this model as log-linear:

$$Pr(T) = \frac{exp(\sum_{(v_1,v_2,l)\in T} \theta^t \cdot \Phi(v_1, v_2, l))}{Z(V, \theta)}$$

- The gradient of the log-likelihood is given by:

$$\frac{\partial LL}{\partial \theta} = \sum_{i=1}^{N} \Big[\sum_{(v_1,v_2,l)\in T_i} \Phi(v_1, v_2, l) - \mathbf{E}_T\big(\sum_{(v_1,v_2,l)\in T} \Phi(v_1, v_2, l)\big)\Big]$$

- **גישה נוספת -** $Stracture\ Perceptron$**:**

האלגוריתם מעדכן את הגרדיאנט בעזרת החסרת סכום הפיצ'ר פאנקשיין על כל הצלעות שנמצאות בעץ שממקסם את ה $score$.

MST Parser: Inference and Learning

1. $\theta^{(0)} \leftarrow 0$
2. **for** $r = 1 \ldots N_{iterations}$
3. **for** $i = 1 \ldots N$
4. $T' \leftarrow argmax_T \sum_{(v_1,v_2,l)\in T} score_\theta(v_1, v_2, l)$
5. $\theta^{((r-1)N+i)} \leftarrow \theta^{((r-1)N+i-1)} + \eta \cdot \left(\sum_{(v_1,v_2,l)\in T_i} \Phi(v_1, v_2, l) - \sum_{(v_1,v_2,l)\in T'} \Phi(v_1, v_2, l) \right)$
6. **return** $\frac{1}{N \cdot N_{iterations}} \sum_k \theta^{(k)}$

T_i is the gold standard tree

Learning Rate

Feature function of T' instead of expectation over T!

לבסוף נחזיר את **ממוצע** הפרמטרים.

7.2.2 $Higher\ order\ models$ - מודלים שמשתמשים בהערכה - $approximate$:

- **המודל** $grandchild$: הוא לא מתפרק לצלעות בודדות, אלא ה $score$ של העץ הוא סכולם לפי שלשות של קודקודים. והפיצ'רים הם על קשר בין מספר קודקודים ולא רק על צלע בודדת.
  ייתרון: זה שימושי כשיש לנו שתי צלעות שכל אחת מהן בפני עצמה סבירה או לא, אך יחד הן מייצרות שרשרת סבירה.

$$score(T) = \sum_{\{(i,j,k)\in V^3 \mid (i,j)\in T \ \&\ (j,k)\in T\}} score(i,j,k)$$

$$score(i,j,k) = \theta^t \cdot \Phi(i,j,k,x_{1:n})$$

- **מודל נוסף** - $sibling$: ה $score$ העץ יתפרק לסכום על פני שלישיות. אך הפיצ'ר יוגדר על שתי צלעות סמוכות בלבד.
  ייתרון: שימושי כשיש לנו פועל עם שני שמות עצם. כך נוכל להסתכל על הצלעות ולראות כי שתיה יחד לא סבירות.

- The sibling model with labels would look like this:

$$score(T) = \sum_{(i,j,k)\in V^3 | (i,j)\in T \& (i,k)\in T} score(i,j,k)$$

$$score(i,j,k) = \theta^t \cdot \Phi(i,j,k,x_{1:n})$$

- Inference (finding the highest scoring tree given θ) is NP-complete. However, there are approximate algorithms.

7.3 $Transition\ Base\ Parsing$:

- **נוטציה:** יש לנו משפט, אנו מגדירים אוסף קודקודים שהן מילים משמשפט, בתוספת קודקוד מיוחד שיסומן כשורש, שלו אין הורה. לכל שאר הקודקודים יש הורה. נסמן אותם ב $0-n$. אנו נרצה לחזות את הצלעות בין הקודקודים.

- **הרעיון:** אנו לא ננסה לחזות את העץ ישירות כאובייקט קומבינטורי היררכי, אלא נרצה לחזות סדרה של פעולות שיובילו אותנו לעץ.
  אנו עושים זאת כי חיזוי של סדרת פעולות קלה יותר מאשר חיזוי שלעץ מלא.

- **מימוש:** נחזיק מחסנית, ונגדיר אוסף פעולות. כך שאם נבצע אתהפעולות אחת אחרי השניה לבסוף נקבל עץ.

- **הגדרת המעברים האפשרים:** בכל נקודה יש נו את המצב שבו אנו נמצאים - קונפיגורציה שיש בה את הדברים הבאים:
  המחסנית - Σ: קודקודים שאנו עדיין משתמשים בהם.
  באפר - B: סדרת המילים שאנו יכולי לקרוא אחת אחרי השניה.
  אוסף צלעות שנחזו עד כה A: אוסף צלעות שמתחיל מעץ ריק ומסתיים בעץ מלא. (בכל שלב באמצע יש לנו תת עץ).

- $Transision\ system$: מורכבת מהרביעיה הבאה:

$$S = (C, T, c_s, C_t)$$

  והיא:
  C - אוסף של קונפיגורציות אפשריות של הפארסר.
  T - אוסף של פונקציות של מעבר בין קונפיגורציות.
  c_s - פונקיה שממפה מצב התחלתי .
  C_t - קונפיגוריציות סיום, כשנגיע אליהן נחזיר את הפלט.

- **סדרה של** $Transision$: מקביל לאאוטפוט.

Transition sequence

A *transition sequence* is $(c_0, \ldots, c_m) \subseteq \mathcal{C}$ s.t.
- $c_0 = c_s(w)$
- $c_m = (\Sigma_m, B_m, A_m) \in C_t$
- For each $i = 1, \ldots, m$ there exists $t \in \mathcal{T}$ s.t. $c_{i+1} = t(c_i)$.

The output of the system is then $T = (V_w, A_m)$.

7.3.1 מערכת $Arc\ standart$:

- **הגדרת המערכת:**

Arc-standard transition system (Nivre, 2004)

Transition set $\mathcal{T}$:

SHIFT	move one item from the buffer to the stack: $(\Sigma,\ i\|B,\ A) \Rightarrow (\Sigma\|i,\ B,\ A)$
LEFT-ARC$_\ell$	create arc $s_0 \to s_1$ with label $\ell \in \mathcal{L}$ and remove s_1: $(\Sigma\|i\|j,\ B,\ A) \Rightarrow (\Sigma\|j,\ B,\ A \cup \{(j, \ell, i)\})$ Condition: $i \neq 0$
RIGHT-ARC$_\ell$	create arc $s_1 \to s_0$ with label $\ell \in \mathcal{L}$ and remove s_0: $(\Sigma\|i\|j,\ B,\ A) \Rightarrow (\Sigma\|i,\ B,\ A \cup \{(i, \ell, j)\})$

Typically $|\mathcal{L}| \approx 50$, so there are 101 different transitions.
Initial configuration:

$$c_s(w_1, w_2, w_3, \ldots,) = ([0], [1, 2, 3, \ldots], \emptyset)$$

Terminal configuration:

$$c_t = ([0], [], A)$$

הפעולות האפשריות:

- $shift$: קריאת מילה מהבאפר והכנסת המילה למחסנית.

- $left\ arc_l$: נייצר צלע עם התוית l ע"י לקיחת שתי המילים הראשוניות במחסנית, ונייצר צלע מהראשונה לשניה (מהעליונה לתחתונה) ונזרוק את המילה השניה i מהמחסנית.
  נבצע את השלב הזה רק אם i (המילה שנזרוק) עלה, או אם כבר בנינו את תת העץ הזה.

- $right\ arc_l$: נייצר צלע עם התוית l ע"י לקיחת שתי המילים הראשוניות במחסנית, ונייצר צלע מהשניה לראשונה (מהתחתונה לעליונה) ונזרוק את המילה הראשונה j מהמחסנית.
  נבצע את השלב הזה אם j עלה או אם כבר ייצרנו את תת העץ הזה.
- **הגדרה - קבוצת עצים פרוייקטיבים:** עצים שבהם אם נצייר את כל הקשתות מצד אחד (מעל או מתחת), לא יהיו לנו צלעות נחתכות.
  במילים אחרות: כל סדרת קודקודים של תת עץ, היא תת סדרה רציפה של המשפט.
- **תכונות:**
  1: כל סדרת מעברים חוקיים יוצרת עץ פרוייקטיבי.
  2: כל עץ פרוייקטבי ניתן לחזות אותו ע"י איזשהי סדרה.
  3: סדרה של מעברים ליצירת עץ בגודל n קודקודים תהיה תמיד באורך $2n$.
  4: חיבור קודקוד נעשה רק אחרי שחיברנו את כל התלויות שלו (ברגע שהוא יוצא מהמחסנית הוא נמחק.)

7.3.2 מערכת $Arc-eager$:

- **הגדרת המערכת:**

Arc-eager transition system (Nivre, 2004)

SHIFT (same)	move one item from the buffer to the stack: $(\Sigma,\ i\|B,\ A) \Rightarrow (\Sigma\|i,\ B,\ A)$
LEFT-ARC$_\ell$	create arc $b_0 \rightarrow s_0$ with label $\ell \in \mathcal{L}$ and remove s_0: $(\Sigma\|i,\ j\|B,\ A) \Rightarrow (\Sigma,\ j\|B,\ A \cup \{(j,\ell,i)\})$ Condition: $i \neq 0$ and i has no head
RIGHT-ARC$_\ell$	create arc $s_0 \rightarrow b_0$ with label $\ell \in \mathcal{L}$ and shift b_0: $(\Sigma\|i,\ j\|B,\ A) \Rightarrow (\Sigma\|i\|j,\ B,\ A \cup \{(i,\ell,j)\})$
REDUCE	remove s_0: $(\Sigma\|i,\ B,\ A) \Rightarrow (\Sigma,\ B,\ A)$ Condition: i has a head

הפעולות האפשריות:

- $shift$: קריאת מילה מהבאפר והכנסת המילה למחסנית.
- $left\ arc_l$: נייצר צלע עם התוית l ע"י לקיחת שתי מילים - המילה הראשונה בבאפר, והמילה הראשונה במחסנית, ונייצר צלע מהראשונה (המילה בבאפר) לשניה (המילה במחסנית) ונזרוק את המילה שנמצאת במחסנית.
  נבצע את השלב הזה רק אם i (המילה שנזרוק) עלה, או אם כבר בנינו את תת העץ הזה.
  ההבדל מהמערכת הקודמת: ניתן למתוח את הצלע מהבאפר ישירות מבלי לעשות לה $shift$ למחסנית.
- $right\ arc_l$: נייצר צלע עם התוית l ע"י לקיחת שתי מילים - המילה הראשונה במחסנית, והמילה הראשונה בבאפר. ונייצר צלע מהשניה (המילה במחסנית) לראשונה (הראשונה בבאפר) . לאחר מכן המילה שהייתה בבאפר תימחק מהבאפר ותעבור למחסנית.
  ההבדל מהמערכת הקודמת: המילה נכנסת למחסנית רק אחרי החיבור. בנוסף היא לא נמחקת אלא עוברת לבאפר.
- $Reduce$: נוציא מהמחסנית מילה, אם יש לה כבר הורה.
- **סיבוכיות:** עבור n מילים יהיו $2n$ חזרות במקסימום סה"כ.
- **תכונה:** אנו יכולים ליצור עצים פרוייקטיבים בלבד.

7.3.3 הוספת פעולת $swap$:

- נרצה לייצר עצים שאינם פרוייקטיבים, אך שתי המערכות שהצגנו אינן יכולות לייצר עצים כאלו. לכן הפתרון הוא להוסיף פעולה נוספת הנקראת $swap$. פעולה זו מוציאה את המילה מהמחסנית ומעבירה אותה לבאפר.
- כך נוכל לייצר כל עץ בסיבוכיות זמן של $O(n^2)$

7.4 כיצד נעשה $Transition - Base\ Parsing$:

7.4.1 אפשרות ראשונה - אלגוריתם חמדן:

- **הרעיון:** נפתור את הבעיה הבאה, אנו נרצה לחזות את האובייקט הבא בסדרה, בהינתן העץ עד כה, המחסנית והבאפר

$$P\left(t_1,\ldots,t_m \mid w\right)=\prod_{i=1}^{m} P\left(t_i \mid t_1,\ldots,t_{i-1},w\right)=\prod_{i=1}^{m} P\left(t_i \mid c_{i-1}\right)$$

למעשה נחפש את המקסימום הבא:

$$argmax_{t_1\ldots t_m\in T}\prod_{i=1}^{m} P\left(t_i \mid c_{i-1}\right)$$

אך מכיוון שאין לנו את סדרת הפעולות, נצטרך להשתמש באורקל שתתן לנו את סדרת הפעולות בהינתן עץ:

$$o(T)=(t_1\ldots t_m)$$

כיצד ניצור את האורקל:

Oracle for arc-standard

while $B \neq []$ and $\Sigma \neq [0]$ **do**
  if $s_0 \xrightarrow{\ell} s_1$ and s_1 has all its children and $s_1 \neq 0$ **then**
    return LEFT-ARC$_\ell$
  else if $s_1 \xrightarrow{\ell} s_0$ and s_0 has all its children and $s_0 \neq 0$ **then**
    return RIGHT-ARC$_\ell$
  else
    return SHIFT
  end if
end while

איך נחזה:

- **אלגוריתם חמדן** אפשרות אחת היא ללמוד בצורה חמדנית עבור כל קונפיגורציה, בכל שלב נקח את המצב המקסימלי מבלי להתחשב בהשלכות.

Greedy transition-based parsing

In *greedy parsing*, instead of

$$(t_1,\ldots,t_m)=\underset{t'_1,\ldots,t'_m\in\mathcal{T}}{\arg\max}\prod_{i=1}^{m} P(t'_i|c_{i-1})$$

we select each transition separately and sequentially:

$$t_i=\underset{t'_i\in\mathcal{T}}{\arg\max}\, P(t'_i|c_{i-1}) \quad i=1,\ldots,m$$

A score $s(t,c)$ estimates this probability. Parsing algorithm:
  $c \leftarrow c_s(w)$
  while $c \notin C_t$ **do**
    $c \leftarrow \left(\arg\max_{t\in\mathcal{T}} s(t,c)\right)(c)$
  end while

יעילות: האלגוריתם הזה מצליח לחזות באופן יעיל. מכיוון ששפה משמרת מודל מסויים.

נחפש את ה $score$ **המקסימלי כך:**

Transition classifiers

Learn the score giving maximum probability to oracle transitions:

$$\arg\max_{s\in\mathcal{S}} \sum_{i=1}^{m} s(t_i^*, c_{i-1}^*)$$

where $t_1^*, \ldots, t_m^*$ (and $c_1^*, \ldots, c_m^*$) are determined by the oracle.

Possible hypothesis classes $\mathcal{S}$:

1. Linear (perceptron, SVM)
2. Feedforward neural networks
3. Recurrent neural networks (RNN, LSTM, GRU)

And others

ונחזה את הפונקציה הבאה:

$$\textbf{Given features } \mathbf{f} = (f_1, \ldots, f_K) : \mathcal{C} \to \mathbb{R}^K \textbf{, learn weights } W_{|\mathcal{T}|\times K} :$$

$$s(t, c) = [W \cdot \mathbf{f}(c)]_t$$

המימוש הטוב ביותר הוא באמצעות RNN עם מודל $LSTM$.

טעויות: אם נעבוד בצורה חמדנית יכולות להיווצר לנו טעויות של צלעות לא נכונות. בנוסף המילה תמחק ולא נוכל לתקן את הצלע.

7.4.2 $Beam\ Seach$:

- **הרעיון:** במקום לשמור בכל שלב את הקונפיגורציה באופן חמדני. נחזה בכל פעם את k הקונפיגורציות הטובות ביותר (מעין תכנון דינמי).
נחזיק תור שיש בו קונפיגורציות, ובכל שלב נסתכל על כל הקינפיגורציות שיש בתור ונבחר עבור כל אפשרות את k האפשרויות הבאות. ולכל סדרה ניתן את ה $score$ שהוא סכום עד כה. לאחר מכן נזרוק את הקודמות ונישאר עם k האחרונות.

- **האלגוריתם:**

Beam search algorithm

Maintain beam Q (of maximum magnitude k) of top-scoring configurations with their scores:

$Q \leftarrow \left\{\left(c_s(w),\ 0\right)\right\}$
while there exists $(c, s) \in Q$ s.t. $c \notin C_t$ **do**
  $Q \leftarrow \textsc{select}\left(k,\ \left\{\left(t(c),\ s + s(t, c)\right) \mid (c, s) \in Q, t \in \mathcal{T}\right\}\right)$
end while
return $\textsc{select}(1, Q)$

- **התמודדות עם טעויות:** בשונה מהאלגוריתם החמדן, כאן אם עשינו צעד לא נכון, אנחנו נוכל לתקן כי בשלב מסויים לא נוכל להמשיך עם הצעד שהיה טעות, ונבחר את הצעד הנכון.

7.4.3 *Training with exploration*:

- **הרעיון:** נשנה את ה *training*. הגישה היא לא הבעיה שהצלע הלא נכונה נבחרה, אלא שתהיה לנו טעות נגררת. לכן עדיין נבצע חיפוש חמדני, אך נטעה בכוונה, כדי להכניס רעש. נעשה זאת באמצעות *dinamic oracles*.
- *dinamic oracles*: אם יש מסלול למצבי הסיום יחזיר את המסלול הזה. אך אם אין מסלול למצבי הסיום, הוא יחזיר סדרת מעברים שהכי קרובה לסדרה המקורית. והכי קרובה לעץ T הרצוי.

8 *Information Extraction*:

מעבר ממידע לא מובנה למידע מובנה, שיישב בטבלאות שניתן לעבוד איתן. כלומר, נרצה לתרגם טקסט לטבלת מידע. כך נוכל לעבוד עם המידע באופן טוב יותר.

השלבים:

1. **נבצע** NER: נזהה עבור את השמות, ולאחר מכן נזהה אם הם חברה, אדם, מקום ועוד.
2. **זיהוי יישויות:** נזהה אם ישות אחת חוזרת מספר פעמים בטקסט, ונשייך אותן אחת לשניה.
3. **זיהוי יחסים:** נזהה את היחסים בין הישויות בטסט, האם הוא בין 2,3 או יותר ישויות (יצרו קשר בניהם, או חברת בת ועוד).

8.1 איך נגדיר את אוסף היחסים - information extraction:

8.1.1 גישה מבוססת אונטולוגיה:

הגישה: יש מבנה היררכי מוגדר מראש של יחסים, עם סוגי היחסים וכל אחד מהם מחולק לתת סוגים. אם יש יחס בטקסט שלא מופיע במבנה אזי הוא לא רלוונטי.

חילוץ המידע מהטקסט: נרצה דרך יעילה לחלץ את היחסים מהטקסט. נדגים על דפי ויקיפדיה, שלהם יש עמודת יחסים ולכן קל יותר לדעת איזה יחס לחפש.
נעבור על היחס בעמודת היחסים, ולאחר מכן נחפש אותו בטקסט. לדוגמה עבור היחס בראד פיט וג'ניפר אניסטון נשואים, נחפש משפט המכיל את השמות של שניה ונראה אם מופיע שם יחס נישואים.
נשים לב: כי שיטה זו מכילה מלא רעש, כי אנחנו נגיע למשפטים שיש בהם את השמות של שניהם, אך המשפט לא מדבר על נישואין, אלא למשל על חברה שהם פתחו יחד.

האלגוריתם:

Schematized Algorithm for Distant Supervision

```
function DISTANT SUPERVISION(Database D, Text T) returns relation classifier C

  foreach relation R
    foreach tuple (e1,e2) of entities with relation R in D
      sentences ← Sentences in T that contain e1 and e2
      f ← Frequent features in sentences
      observations ← observations + new training tuple (e1, e2, f, R)
  C ← Train supervised classifier on observations
  return C
```

8.1.2 גישה שניה - open information extraction:

גישה שניה - open information extraction: אין אוסף יחסים מוגדר מראש, אלא אנחנו מגדירים את היחסים ע"י מילים שמופיעות בטקסט.

חסרונות:

1. **בעיה ראשונה:** מופעים של אותו היחס לא יוגדרו אותו היחס אם הופיעו מילים שונות בטקסט כדי לתאר את היחס.
2. **בעיה נוספת:** המילים שמגדירות את היחס לפעמים מגדירים יחס שאנחנו לא רוצים להגדיר בדרך הזו, כי הם נדירים או שהם לא מספיק אבסטרקטים.
   לכן נרצה יחסים סמנטים ולא ניסוחים תחביריים.

תיוג תפקידים סמנטים - Semantic Role Labeling (SRL): פתרון הבעיה. נגדיר את היחסים בצואה אבסטרקטית. בהינתן משפט נזהה לכל יחס מה הארגומנטים (משתתפים) ומה התפקיד של כך אחד מהם, את התפקידים הסמנטים נקח מאונטולוגיה. כך יהיה נו מבנה על המשפט, אך הוא לא יהיה מבנה תחבירי.

יש שתי שיטות לביצוע SRL:

1. $FrameNet$: הרעיון בבסיב השיטה הוא שלפני שנתאר את התפקידים הסמנטים נגדיר מהם סוגי האירועים.

חילוץ המידע מהטקסט: נקח טקסט ונבצע עליו ניתוח תחבירי לעץ. באמצעות העץ נסמן את הארגומנטים הפוטנציאלים, נזרוק את שאר הקודקודים, נעבור על הקודקודים שנשארו ונבצע עליהם בדיקות.

8.2 ניתוח זמן מטקסט:

נרצה תיאור זמן של הטקסט, מה קרה מתי, מה לפני מה, באיזו תקופה ועוד.

דרך אחת לייצוג: נייצג את היחסים באמצעות אלגברה. אך עדיין יש לנו מספר קשיים כיצד לייצג בצורה הטובה ביותר.

9 תרגום אוטומטי - Machine Translation:

הבעיה: נרצה לדעת אם מדובר בטקסט או בנאום, משפט, פסקה, מסמך. האם נרצה תרגום אחד או מספר תרגומים. האם הטקסט ספרותי ועוד.

ראשית נצטרך להפיג עמימות, שיהיה ברור לנו מה הפועל בא להגיד בשפת המקור. לדוגמה הפועל $play$ יכול להיות מתורגם ל "משחק" ו "מנגן".

בעיות איתן מערכות תרגום צריכות להתמודד:

1. סדר המילים - היכן נמקם את הפועל, ומה סדר הנשוא והמושא.
2. שפות שבהן אין כינויי גוף. לדוגמא "הלכתי הביתה", בתרגום נצטרך להוסיף כינוי גוף.
3. שפות ללא הטיות זמן, בסינית אין ביטויי זמן ונצטרך להבין ולתרגם עם הטית הזמן הנכונה.
4. Lexical Gaps: ישנם ביטויים שקיימים בשפה אחת אך לא קיימים בשפה אחרת.

רעיון שפת Inter lingua: שפה סמנטית אבסטרקטית. עלה רעיון לתרגם את שפת המקור לשפת Inter lingua, ולאחר מכן לשפת היעד.

9.1 איך נעשה $evaluation$:

ראשית הבעיה היא בעיה קשה, כי משפט אחד ניתן לתרגם למספר משפטים שונים. בנוסף זה לא ראלי לקחת מתרגמים שיבדקו מה נכון. נרצה להשתמש בתרגומי רפרנס שימדדו לנו את איכות התרגום.
העבודה המשפיעה ביותר על התחום היא $BLEU$.

9.1.1 מודל $BLEU$:

הרעיון: נחשב סוג של מדד ומתקנת אותו.

1. תחילה נחשב $unigram$, כמה מהמילים שמופיעות במשפט המתורגם מופיעות גם בלפחות אחד מהרפרנסים.
2. לאחר מכן נחשב $bigram$, כמה זוגות של מילים מופיעות לפחות באחד מה רפרנס.
3. אח"כ נחשב

$$p_n = \frac{\sum_{C\in \text{ corpus}} \sum_{\text{n-gram }\in C} \text{ count }_{\text{ match }}(\text{n}- \text{ gram })}{\sum_{C\in \text{ corpus}} \sum_{\text{n-grameC}} \text{ count }(\text{n}- \text{ gram })}$$

הערה: אם מילה אחת הופיעה בתרגום n פעמים, וברפרנס היא הופיעה $n-1$ פעמים. נספור כאילו היא הופיעה $n-1$ פעמים, למרות שהיו n התאמות.
חסרון: אם מודדים כמה המודל מדוייק, המודל יוציא רק מה שהוא בטוח בו, למרות שהוא לא נכון. כי הוא יודע שזה מה שנמדד. לדוגמה הוא יוצא $4-gram$ אחד נכון ויקבל ניקוד גבוה. כך תהיה לו נטייה להוציא משפטים קצרים.

9.1.2 מודל Brevity Penalty:

הרעיון: נוסיף גורם נוסף שמעניש על תרגגומים קצרים, כדי להימנע מהבעיה שראינו קודם.
נעשה זאת כך:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

כאשר r הוא האורך הרצוי, ו c הוא האורך שהמודל הוציא.

9.2 הגישה הקלאסית - שיטות סטטיסטיות:

בעיות של שיטות סטטיסטיות:
Word Alignment: למרות שיש לנו *corpus* עם משפטים מתורגמים, עדיין אנחנו נצטרך לדעת איזה מילים מתורגמות לאיזה מילים במדוייק. כדי שנוכל להשתמש טוב בסטטיסטיקה. בנוסף יכול להיות שיהיו מילים שאין להם תרגום לשפת היעד, או מילה שיש לה תרגום למספר מילים במשפט בשפת היעד.

9.2.1 מודלי IBM:

הרעיון: שיטה סטטיסטית, באמצעות Direct transfer נתרגם סטרינג לסטרינג.

1. נתרגם כל משפט בשפת המקור לאחר שנעשה לו אנליזה מרפולוגית, אותו הדבר נעשה למשפטים בשפת היעד. כלומר - ניפטר מההטיות של המילים, לדוגמה "ירוקה" תהפוך ל - "ירוק": נקבה.
2. לאחר מכן נתרגם את המילים.
3. אח"כ נסדר מחדש את המילים לפי הסדר בשפת היעד.
4. לאחר מכן נחזיר את ההטיה המקורית.

חסרונות: זה עבד רק על שפות שיחסית דומות אחת לשניה.

9.2.2 שיטות מבוססות תחביר:

הרעיון: נשתמש בעץ התחבירי של השפות. וננסה ללמוד באמצעות שיטות סטטיסטיות איך לשנות את העץ של שפת המקור כך שיתאים לעץ של שפת היעד.

כלומר: למרות שסדר המילים שונה בין שפות, אנחנו יכולים לנסח את ההכללה באמצעות עצים. כך ההכללה תהיה פשוטה עד כדי החלפת בנים של אותו קודקוד בעץ. בעקבות כך התפתחו גישות שממפות בין המבנה התחבירי של שפה אחת למבנה התחבירי של שפה אחרת. אך זה עדיין לא מספיק כי לפעמים יש הבדלים מהותיים בין משפטים בשפות שונות.

9.3 שיטות מבוססות רשתות נוירונים:

9.3.1 שיטה ראשונה - encoder decoder:

הרעיון: ארכיטקטורת מצפין ומפענח. אנחנו מכניסים רצף מילים וקבלים תרגום.

- **ה** $encoder$ - מקבל את כל המילים בשפת המקור, והוא צריך לקודד את כל המשפט לווקטור שנקרא $summury\ vector$, הוא מייצר ווקטור כזה עבור כל מילה במשפט, באמצעות רשת RNN. לבסוף הוא מחזיר ווקטור שמייצג את סוף המשפט end of sentence שבו מקודדות כל המילים במשפט וההשפעות שלהן על המשפט.

- **ה** $decoder$ - רשת RNN נוספת, היא מקבלת את הווקטור end of sentence ומייצרת את המילה הראשונה בתרגום (הפלט הוא $softMax$ עם הסתברות לכל מילה ב $corpus$). אח"כ נקח את המילה הראשונה ונכניס אותה כאינפוט ל $decoder$ עם שאר הווקטור. נחזור על השלב הזה עד שנקבל את התרגום לכל המשפט.
  למעשה הוא מקבל בכל שלב את כל המילים שהוא חזה עד כה, עם הווקטור של ה $encoder$.

אימון: נאמן כמו שמאמנים רשת נוירונים רגילה עם $cross\ entrepy\ loss$. נרצה למקסם את log ההסתברות של הטוקן הבא, בהינתן הטוקנים הקודמים.

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\log\left[p_\theta\left(y_i \mid y_{i-1},\ldots,y_1,\mathbf{x}\right)\right]$$

החסרון: נהיה דיי קשה לתרגם כאשר המשפט הוא משפט ארוך, כי הווקטור משמש צוואר בקבוק. משום שהוא מייצג את כל המשפט.
בנוסף ווקטור התוצאה של $encoder$ של כל המשפט, אך כך מילה בעיקר תלויה בכמה מילים בודדות במשפט המקורי ולא בכל המשפט. במיוחד אם השפות דומות.
הפתרון: להוסיף מנגנון של Word Alignment.

9.3.2 הוספת Word Alignment בעזרת Adding Attention Mechanism:

הרעיון: לא נעביר את כל המשפט, אלא תהיה לנו פונקציה שתשקול את המצבים השונים של ה $encoder$ בהחלטה של המילה הבאה שאנחנו רוצים לייצרר.
לא נבנה ישירות פונקציית פלט ישירות על end of sentence, אלא נוסיף לו עוד קלט שיהיה צירוף קמור של כל הווקטורים של המילים במשפט. כך לכל מילה תהיה השפעה על המילה המתורגמת.
נלמד כך:

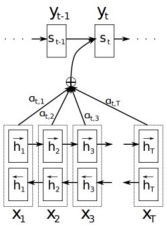
Adding Attention Mechanism

$$p(y_i|y_1,\dots,y_{i-1},\mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \qquad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$



10 Contextualized Word Embeddings:

מוטיבציה: עד עכשיו השתמשנו בדאטה מתוייג, וזה הקשה עלינו משום שאי אפשר לתייג את כל הדאטה בעולם ואנחנו מתייגים מספר סופי של מילים. לכן נרצה להשתמש בדאטה בצורה $unsupervise\ learning$.

הרעיון: נוכל להיעזר במודלי שפה, משום שהם מבוססים על דאטה לא מתוייג. בנוסף כדי לחזות את המילה הבאה צריך לדעת הרבה על המילים הקודמות של המשפט. ומודלי שפה כבר מחזיקים את המידע הזה.

10.1 ארכיטקטורת הרשת:

10.1.1 Self-Attention:

הרעיון: אנחנו נרצה לייצג את המילים באמצעות ווקטורים בצורה הטובה ביותר. נעשה זאת באופן הבא - תחילה ניצור ווקטורים עבור המילים. לאחר מכן נכניס אותם לרשת, כך שבכל שכבה של הרשת הווקטורים ישתפרו וייצגו את המילה באופן טוב יותר.

מבנה הרשת: הרשת תקבל ווקטורים למילים, ותחשב עבור כל ווקטור שלש פרמטרים - $query, key, val$. לאחר מכן עבור כל ווקטור אנחנו נכפול את q שלו ב k של כל שאר המילים, וזה היה המשקל של כמה המילה k משפיעה על המילה הנוכחית. נרכיב מטריצות w_k, w_q, w_v שייצגו את המשקלים הללו. המטריצות הללו הן המטריצות שהרשת תלמד.

סיבוכיות: אנחנו לומדים עבור כל מילה על כל המילים האחרות, סה"כ בכל שכבה עבור n מילים נבצע n^2 חישובים.

10.1.2 ארכיטקטורה של רשת Transformer:

רשת שהארכיטקטורה שלה משמשת באופן דומה למודלים של $encoder - decoder$ -
יש לנו $encoder$ שמקבל ייצוג של המילים, ופרמטר שמייצג את מיקום המילה במשפט - $positional\ Encoding$, ובעזרת בלוקים שלSelf-Attentionמוציא לנו ווקטור שמייצג את המילים בצורה טובה יותר.

לאחר מכן הפלט של ה $encoder$ מחובר ל$decoder$ שתפקידו לחזות את ההתפלגות של המילה הבאה ,בהינתן כל המילים עד כה

$$p_\theta\left(y_i \mid y_{i-1}, \ldots, y_1, \mathbf{x}\right)$$

ה $decoder$ **עובד כך:** מקבל את המילים שחזינו $(prefix)$ עם מיקום המילים במשפט $(positional\ Encoding)$, ולאחר מכן הוא מבצע Cross-Attention- הוא לוקח בחשבון את הייצוג של המילים שקיבלנו עד עשיו, וגם את המשפט המקורי $(sorce\ sentence)$. כלומר בכל שכבה הוא מקבל את הפלט של השכבה הקודמת שזה ה $prefix$ וגם את הפלט של ה $encoder$.
לבסוף הוא מחזיר את ההתפלגות עבור מילה 1 - המילה הבאה.

הפרמטרים של הרשת:

- הפרמטרים של הבלוקים של Self-Attention, מטריצות w_k, w_q, w_v לכל שכבה יש מטריצות משלה.
- הפרמטרים של $forword$ של פספטרון.
- הפרמטרים של האינפוט $word\ to\ vec$, בד"כ נלמד אותן גם כן.

מבנה הרשת בד"כ יהיה עמוק מאד ורחב עם כמה בלוקים של Self-Attention, שכל אחד מהם מייצר כמה ווקטורים ובסוף אנחנו מייצרים מכולם ווקטור אחד טוב.

מיקום המילה במשפט - positional Encoding: יש משמעות למיקום של המילה במשפט, לדוגמה אדם נשך כלב או כלב נשך אדם, יש משמעות שונה למשפט, לכן סדר המילים במשסט חשוב לנו. במודל של Self-Attention אין משמעות לסדר המילים, לכן נכניס לרשת ה $Transformer$ ייצוג של המיקום של המילה שיילקח בחשבון, הייצוג של המילה יתחשב באינדקס של המילה במשפט, אך לא בסדר של המילים במשפט.
הפרמטר הזה הוא גם כן נלמד.

יתרונות: בשונה מרשתות RNN שבהן עשינו $back\ prop$. ברשת הזאת כל שכבה לא ממש תלויה בשכבה הקודמת, לכן ניתן לאמן אותן במקביל.
בנוסף, אין לו חשיבות מסויימת למיקום המילה במשפט, אלא הוא מתייחס מילה כאינדקס, לכן הכל סימטרי והוא חוזה טוב יותר.

10.2 מודל $BERT$:

הרעיון: נקח מודל שמבוסס על $encoder$, ללא ה $decoder$ שלו. ונוסיף טוקן מיוחד שנקרא $masks$, ועליו נבנה $decoder$ שמייצר התפלגות על גבי המסיכות. הוא יחזה הסתברות להשלמת מילה במשפט ממוסך, כלומר ישלים מילים לפי ההסתברות למילים חסרות במשפט.
הרשת לוקחת משפט, או מילה במשפט ומייצר עבורה ווקטור $embeddings$. **היא בנויה על גבי** $Transfomer$ עם $encoder$ -

היא מקבלת משפט בייצוג בסיסי, ומשכבה לשכבה משכללת את הייצוג שלו, כך שהוא מתחשב במילה ובקונטקסט של המשפט.

אימון: נקח טקסט גדול ונכניס אותו ל$encoder$ כדי שיציא לנו ייצוג מדוייק של המשפט. עבור כל טוקן בטקסט נבחר האם לעשות עליו מניפולציה, ב 80% מהמקרים נמסך אותם, ב 10% נחליף בטוקן רנדומלי, ו 10% נחזיר את אותו הטוקן.
המשפט החדש הוא x', ונרצה למקסם את ההסתברות על הלוג של המשפט x הנכון בהינתן x'.
פונקציית ה $Loss$**:**

$$L(\theta) = \frac{1}{|S|} \sum_{i \in S} \log \left(p_\theta \left(x_i \mid \mathbf{x}' \right) \right)$$

איך נשתמש: אם נרצה להשתמש במודל לבעיית NER, נאמן את המודל של $mask$ ואח"כ נעשה לו fine-tuning. נקח את הייצוגים שנמצאים בשכבה הראשונה שלו, ונכניס אותם לרשת חדשה שתבצע NER. נתייחס אליהן כאל רשת אחת.
ההנחה היא כי אם המודל יודע להשלים מילים ממוסכות, הוא כנראה למד מלא דברים על המשפטים - תחביר איך להדלים נכון לוגית ועוד. לכן אנחנו נשתמש בידע הזה למשימות אחרות.

10.3 מודל Zero-shot Prediction:

הרעיון: אם נרצה להוציא מידע מטקסט מסויים. לדוגמה עבור משימת $sentiment\ analisis$ נרצה לדעת האם ביקורת על מוצר מסויים היא חיובית או שלילית. נוכל לכתוב את שם המוצר ולשים מאסק על הביקורת, המודל יקרא את הטקסט ויחזה לנו מה החלק של המאסק.