به نام خدا

تمرین شماره سه درس پردازش زبانهای طبیعی توحید عابدینی ۹۹۷۲۲۱۵۳

سوال ۱)

ابتدا جملات را به صورت زیر علامت گذاری میکنیم تا در محاسبات کارمان راحت تر شود:

Mark | can | watch
N | M | V

Will | can | mark | watch
N | M | V | N

Can | Tom | watch?
M | N | V

Tom | will | mark | watch
N | M | V | N

بخش a)

جدول خواسته شده را تشکیل میدهیم:

با شمارش نقش هر کلمه در حالت های مختلف در جملات احتمال های متفاوت را برای هر کلمه محاسبه میکنیم.

کلمات	Noun	Modal	Verb
mark	1/6	0	2/4
can	0	3/4	0
watch	2/6	0	2/4
will	1/6	1/4	0
tom	2/6	0	0

بخش b)

حال به تشکیل جدول دوم (transitions) پرداخته و آن را تکمیل میکنیم.

برای تکمیل جدول حالت های توالی مختلف را محاسبه میکنیم. یعنی به عنوان مثال سطر Noun و ستون Modal حالت هایی را شامل میشود که پس از Noun یک Modal آمده است. که تعداد آن را میشماریم و تقسیم بر مجموع تمامی حالات آن سطر که شامل تمامی حالاتی است که پس از Noun یک نقش دیگر آمده است. به همین ترتیب جدول را پر میکنیم.

	Noun	Modal	Verb	<E>
<S>	3/4	1/4	0	0
Noun	0	3/6	1/6	2/6
Modal	1/4	0	3/4	0
Verb	2/4	0	0	2/4

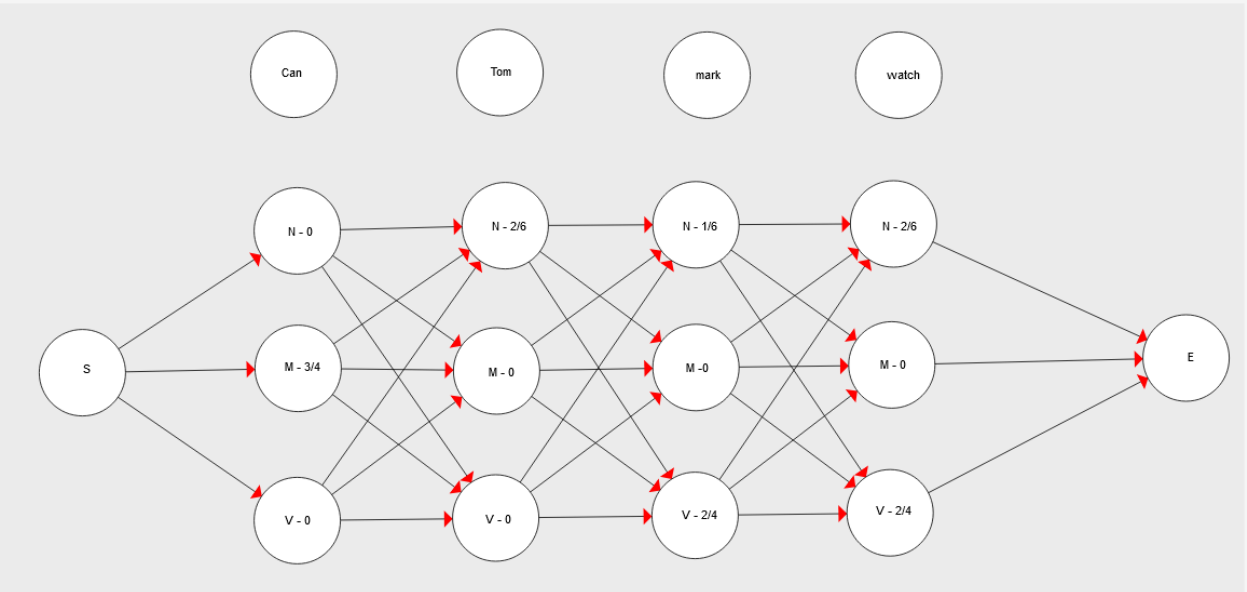
بخش c)

جمله ای که باید تگ زده شود:

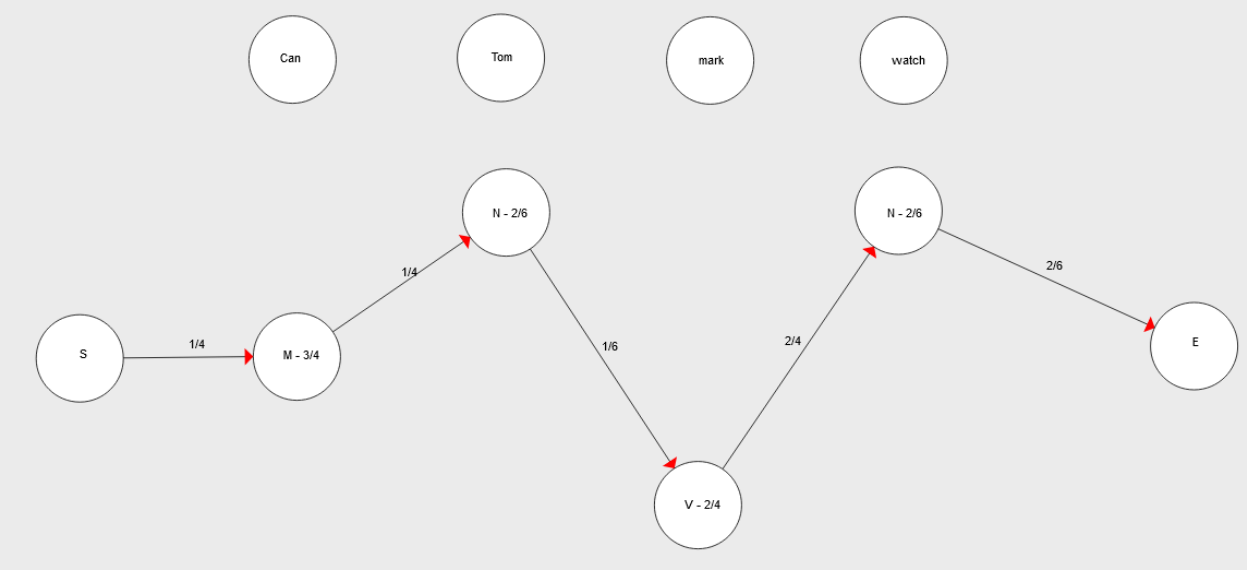
Can Tom mark watch?

شروع به ساختن گراف میکنیم.

در ابتدای گراف ما کامل است و با احتمالات گذار و حالات پر شده است.



حال هم حالات و گذار هایی که احتمال صفر دارند را حذف میکنیم:



به دلیل وجود احتمالات صفر زیاد در حالات گذار گراف نهایی تنها یک مسیر دارد اما اگر چند مسیر داشت باید احتمال را محاسبه میکردیم و ماکسیمم میگرفتیم تا تگ های نهایی را مشخص کنیم.

احتمال مسیر فعلی برابر است با:

$$P(< S >\rightarrow \mathrm{M} \rightarrow \mathrm{N} \rightarrow \mathrm{V} \rightarrow \mathrm{N} \rightarrow < \mathrm{E} >) = \frac{1}{4} * \frac{3}{4} * \frac{2}{6} * \frac{1}{6} * \frac{2}{4} * \frac{2}{4} * \frac{2}{6} * \frac{2}{6} = 0.00028935185$$

و تگ های ما در نهایت به صورت زیر خواهند بود:

Can | Tom | mark | watch
M | N | V | N

سوال ۲)

گرامر داده شده به شرح زیر است:

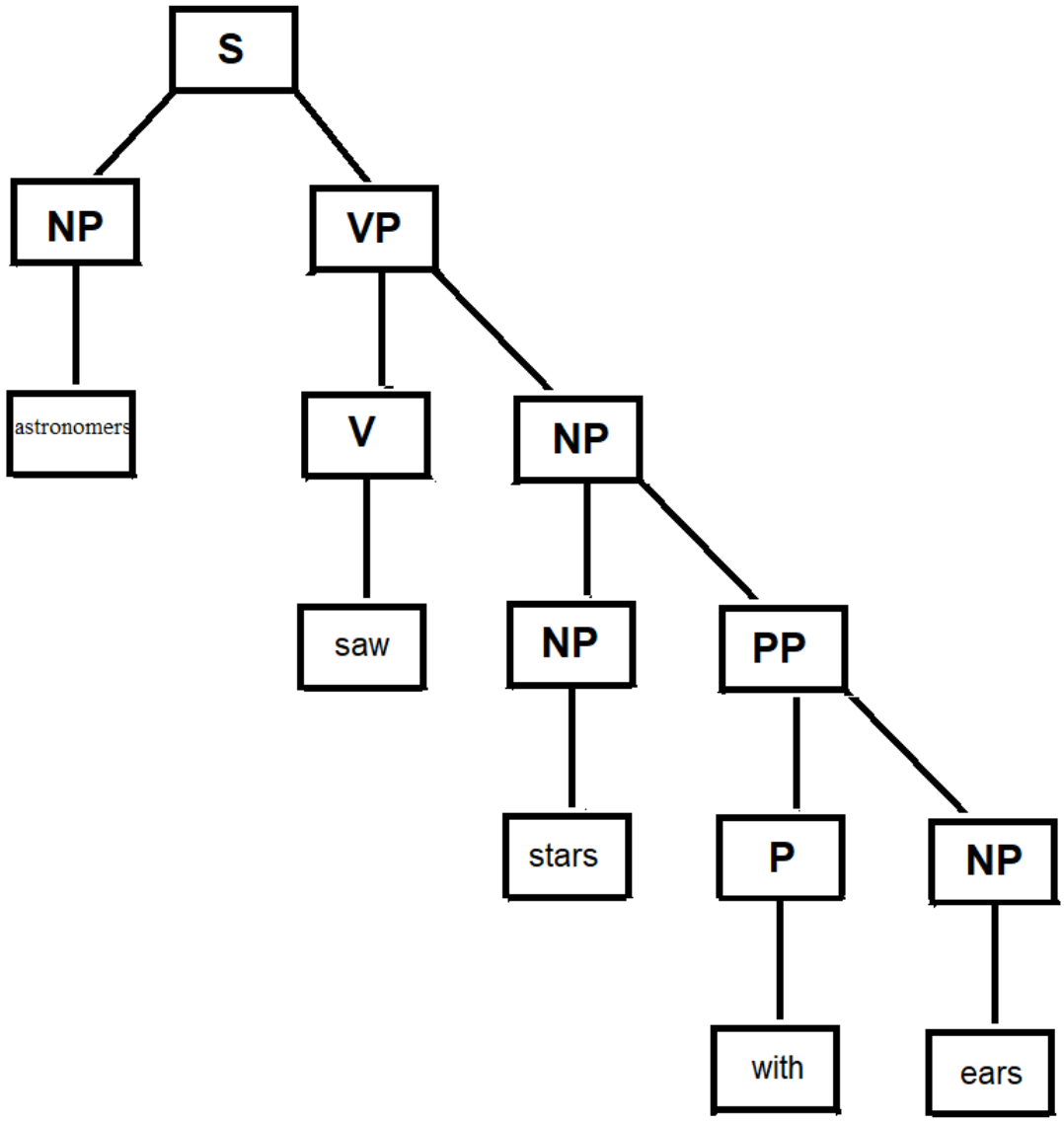
S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → *astronomers*	0.1
VP → V NP	0.7	NP → *ears*	0.18
VP → VP PP	0.3	NP → *saw*	0.04
P → *with*	1.0	NP → *stars*	0.18
V → *saw*	1.0	NP → *telescopes*	0.1

مطابق اسلاید های درس جدول را به صورت زیر ایجاد میکنیم:

ترتیب پر کردن جدول به صورت قطری از پایین به بالا است.
ابتدا ۵ خانه ی موجود در قطر نهایی جدول پر میشنود. سپس به همین صورت به قطر های بالا تر آمده و در صورت امکان خانه های جدید را با تمامی ترکیب های ممکن با خانه های پایین تر پر میکنیم تا به خانه آخر که در این جدول در مرحله ۵ ام (قطر آخر) قرار دارد برسیم.
برای به دست آوردن درخت تجزیه باید به صورت backward عمل کنیم و درخت حاصل را سطح به سطح بسازیم.
برای راحتی مقایسه در هر قطر خانه ها همرنگ شده اند.

astronomers	saw	stars	with	ears
NP → astronomers 0.1		S→ NP VP 0.0126		S→ NP VP 0.0009072
	V → saw 1.0 NP → saw 0.04	VP→V NP 0.126		VP → V NP 0.009072
		NP → stars 0.18		NP → NP PP 0.01296
			P → with 1.0	PP → P NP 0.18
				NP → ears 0.18

درخت تجزیه به صورت زیر است:



سوال ۳)

بخش a)

ابتدایی ترین روشی که به ذهن میرسد جست و جوی به اصطلاح exhaustive روی متن است. مثلا با ریجکس یا تابع find در استرینگ پایتون. همه شروط را بررسی کرده و به صورت Boolean در نهایت AND کرده و نتایج را برمیگردانیم.

بخش b)

روش مذکور مشکلاتی دارد که چند نمونه از آنها عنوان میشود:

- برای متن های طولانی و بزرگ کند عمل میکند.
- Query یا پرسش هایی که حالت منفی و NOT دارند پیاده سازیشان سخت است و به اصطلاح trivial نیست.
- برخی دیگر پرسش ها نظیر اینکه مثلا در نزدیکی یک کلمه به دنبال یک کلمه دیگر بگردیم با این روش جست و جو ممکن نخواهد بود.
- همچنین اگر بخواهیم به صورت رتبه بندی شده کتاب ها را خروجی کنیم و امتیاز دهیم با این روش ساده به مشکل خواهیم خورد.
- این روش از این نظر که برای هر ورودی متفاوت یکبار باید هر کتاب را بررسی کنیم مشکل دارد و اگر ورودی ها تغییر کند باید دوباره تمامی کتاب ها را جست و جو کنیم، مشکل دارد و قابل عملی سازی نیست.

بخش c)

برای ساختاریافته کردن متن و داده در این مساله به این صورت عمل میکنیم:

برای هر کتاب یک id تعیین میکنیم.

یک دیکشنری خواهیم داشت که کلید های آن کلمات ما هستند. یعنی تمامی کلماتی که در تمام کتابها هستند، و مقادیر ما id کتابهایی است که این کلمه در آن ها موجود است (به صورت لیست)

برای ساختن این دیکشنری هر کتاب را ابتدا tokenize کرده، بعد نرمال سازی و سپس stemming و lemmatization میکنیم و stopword ها را حذف میکنیم و در نهایت عمل indexing یا نمایه گذاری را برای آنها انجام میدهیم. به این صورت که کلمات را به صورت الفبایی مرتب کرده (البته مرتب کردن ما دو بخش دارد یکی برای کلمات است که به صورت الفایی بوده و یکی برای ایندکس های هر کلمه مشابه است که به صورت عددی و صعودی است.) و برای کلماتی که در یک کتاب چند بار تکرار شده اند آنها را یکسان کرده و موارد تکراری را حذف میکنیم و سپس یک دیکشنری با استفاده از این لیست دوتایی کلمات و ایندکس کتاب ها ایجاد میکنیم. در هر سطر اعداد مقابل یک کلید به صورت صعودی مرتب شده اند.

شکلهای زیر میتوانند درک بصری بهتری در رابطه با فرآیند ایجاد کند:

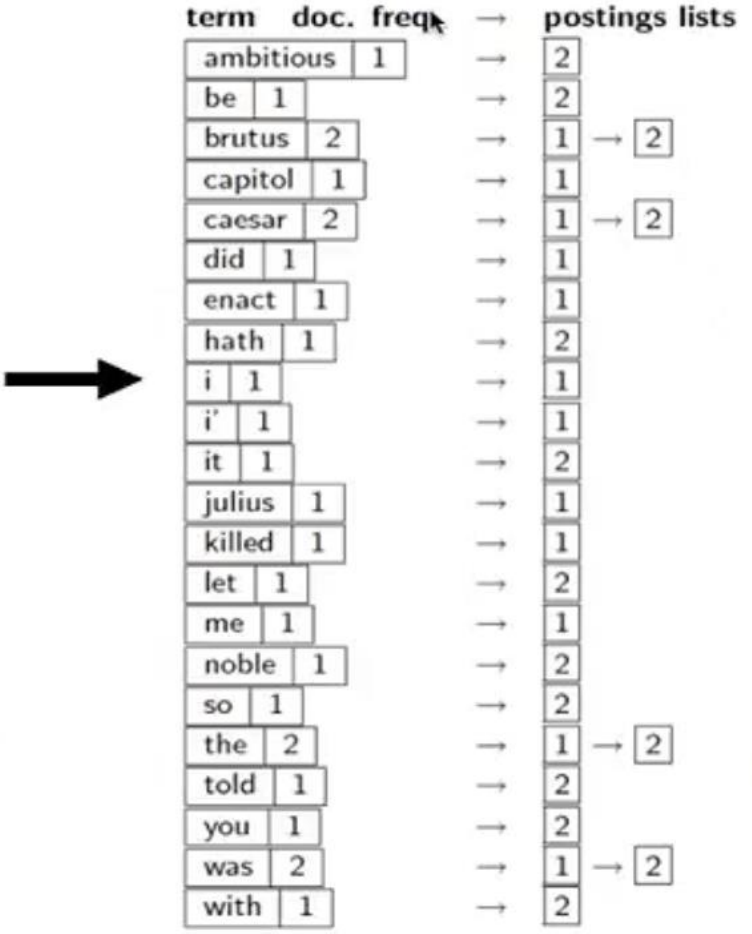
شکل اول بخش مرتب کردن الفبایی و عددی را نشان میدهد:

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

در شکل دوم شاهد حذف کلمات تکراری و ایجاد دیکشنری مذکور هستیم:

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



در کنار هر کلمه تعداد درایه های لیست متناظر با آن نوشته شده است.

و بدین ترتیب داده ما به یک حالت ساختار یافته درآمده است.

بخش d)

در این قسمت برای اجرای یک پرسش یا Query یک فرآیند پیشنهاد میشود:

کلماتی که ما به دنبال اجرای پرسش روی آن هستیم را از دیکشنری یافته و عمل Merge روی آن لیست آن کلمات انجام میدهیم. یا به بیان ساده تر intersection یا اشتراک روی آن ها میگیریم که مشابه عمل AND است که در حالت رویکرد ساده داشتیم. برای حالت NOT هم میتوانیم ایندکس ها را مشابه عمل NOT مجموعه از مجموعه مادر کم کنیم تا به قرینه آن رسیده و سپس عمل Merge را اجرا کنیم.

مجموعه اعداد حاصل id کتابهایی است که شرط پرسش ما را برآورده کرده اند.

منبع:

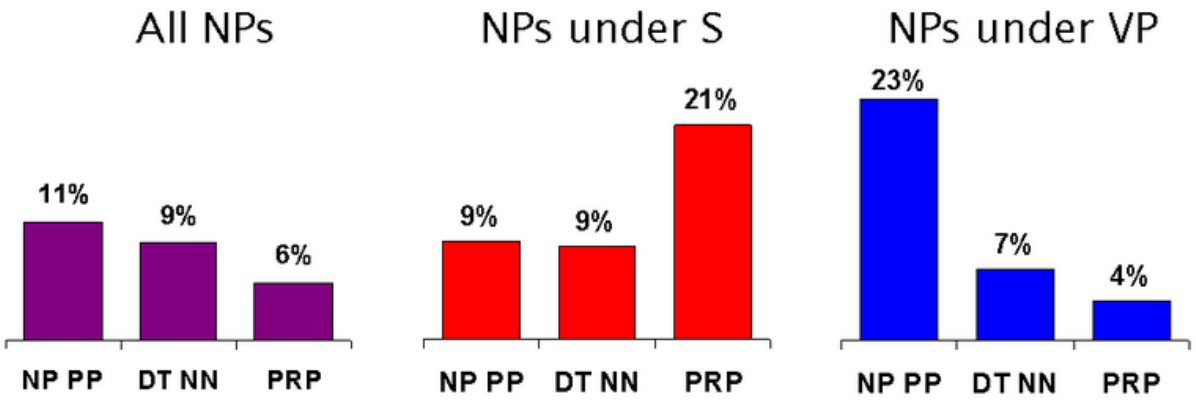
ویدیوی IR دانشگاه استنفورد – آقای Manning

سوال ۴)

بخش a)

مشکل های PCFG عبارت اند از:

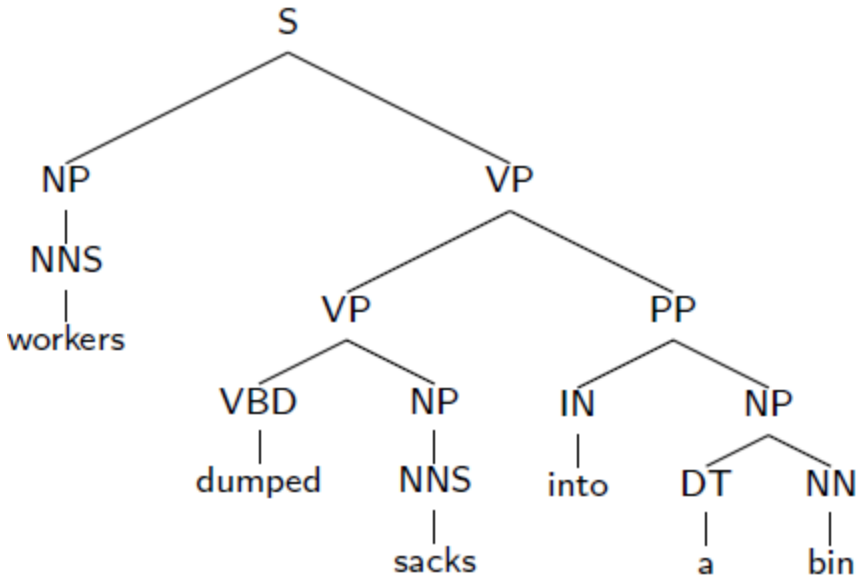
۱) فرض مستقل بودن: قوانین موجود با احتمالاتی محاسبه شده اند که هیچ اهمیتی برای مکان وقوع این قوانین در درخت نمیدهد. فارغ از اینکه یک tag در چه سطحی با چه والد و sibiling ای رخ داده است احتمال مستقل برای آن ها حساب میکند. مثالی که میتواند اشکال این موضوع را برای ما نشان دهد به صورت زیر است: (برگرفته از اسلاید های درس)

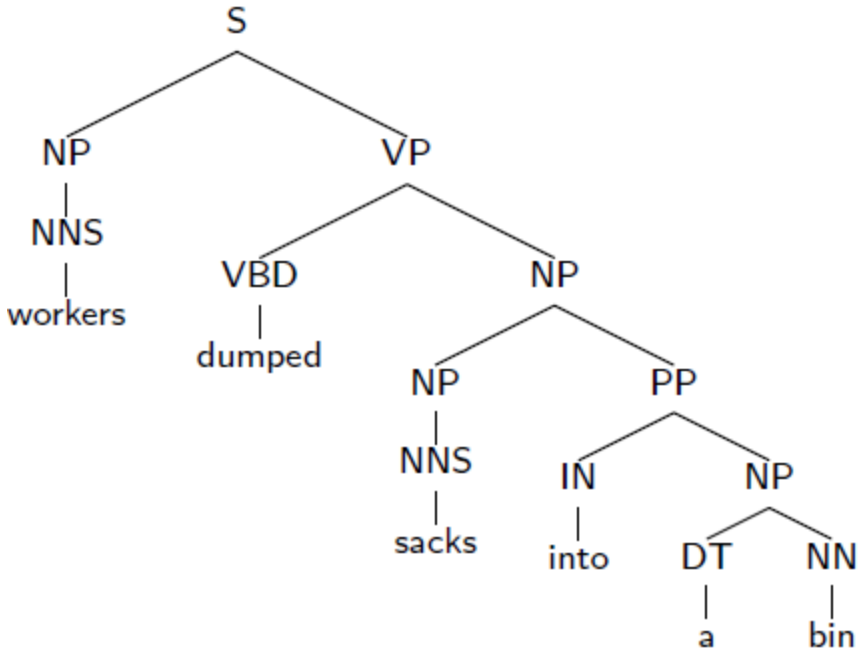


مشاهده میشود که در حالت عادی و مستقل NP های مای در بیشترین حالت به صورت NP PP پارس میشوند اما وقتی NP مذکور دارای والدی با برچسب S باشد این حالت برای بیشترین احتمال به PRP تغییر میکند و اگر هم والد آن VP باشد احتمال NP PP بیشتر میشود و مشاهده میشود که فرض در نظر نگرفتن شروط و برچسب والد و محل وقوع یک head درست نیست و نباید مستقل رفتار کرد (این موضوع ما را دچار خطا خواهد کرد.)

۲) PCFG ها به اطلاعات و وابستگی های lexical یا لغوی حساس نیستند. یعنی برای قوانین اهمیتی ندارد که کلماتی که برای این قانون وجود دارند چه هستند و به بیانی دیگر همه را یکسان در نظر میگیرد. این موضوع هم ما را دچار خطا و ابهام میتواند بکند. یک مثال میتواند این باشد:

برای یک جمله دو درخت صحیح و غلط را میبینیم:





درخت بالایی درست است و درخت پایینی غلط. اما به دلیل اینکه توجهی به کلمات نداریم با یک قانون اشتباه درخت ما اشتباه میشود.

$$NP \rightarrow NP\ PP$$

$$VP \rightarrow VP\ PP$$

قانون های فوق بدون توجه به کلمه ایجاد شده اند و باعث ابهام میشوند.

۳) در PCFG ها یک بایاس به این صورت وجود دارد که احتمال درخت های کوچک تر بیشتر از یک درخت بزرگ تر است.

۴) دو پارس متفاوت اما با مجموعه قوانین یکسان در PCFG (یعنی ترتیب اعمال قانون ها متفاوت باشد) دارای احتمال یکسان است.

بخش b)

با lexicalized گرامر ها این مسائل به این صورت حل میشوند:

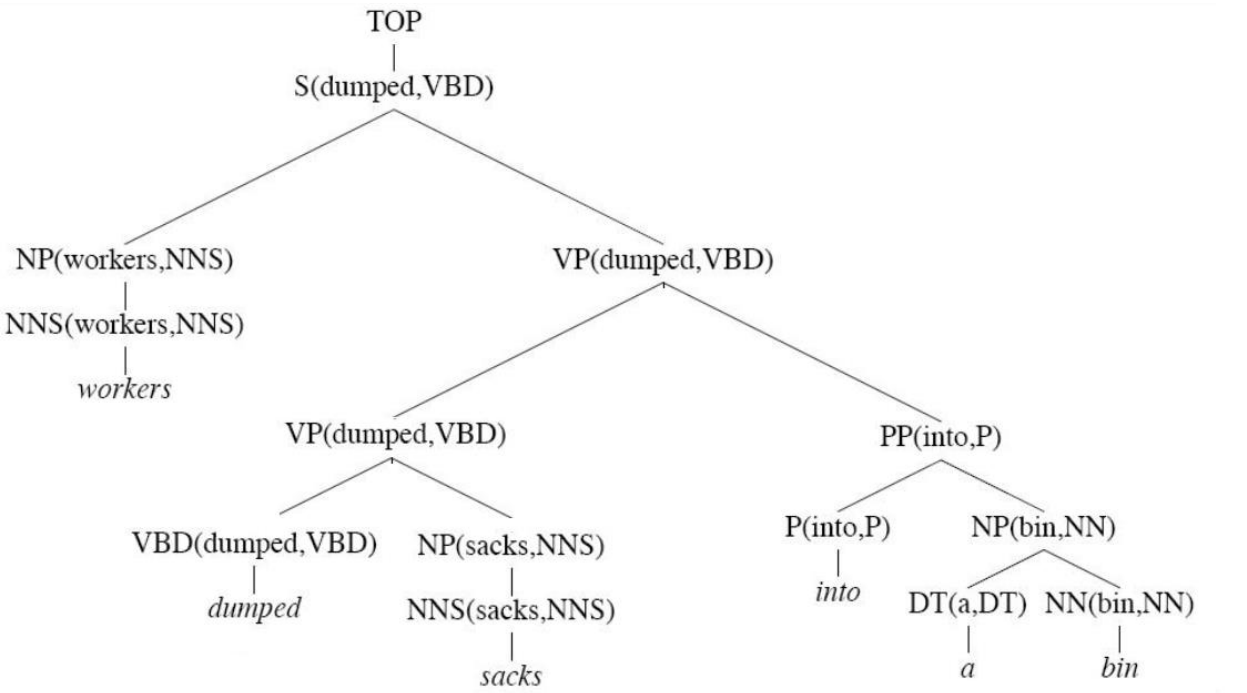
۱) برای حل مشکل فرض استقلال میتوانیم برچسب پدر را به قوانین اضافه کنیم تا یک مرحله وابستگی را گسترش دهیم و برچسب پدر هم در قانون و محاسبه احتمال آن دخیل باشد.

در این حالت قوانین جدید را میتوانیم به این صورت تعریف کنیم:

$$\text{NP-S} \rightarrow \text{PRP}$$
$$\text{NP-VP} \rightarrow \text{NP PP}$$
$$,\ldots$$

که مشاهده میشود اضافه کردن برچسب والد به قانون باعث کاهش این خطا میشود. و اینگونه نیست که برای هر NP بدون توجه به برچسب والد آن گسترش NP PP را به عنوان بیشترین احتمال پیشنهاد دهد.

۲) برای این حالت هم با در نظر گرفتن اطلاعات لغوی میتوان ابهام را رفع کرد. در گرامر های lexicalized علاوه برا برچسب پدر میتوان از کلمه هم به عنوان یک گزینه دیگر استفاده کرد. (البته امکان استفاده تنها از کلمه هم وجود دارد). مشکل موجود در مثال بالا با اضافه کردن کلمه و برچسب والد به صورت زیر قابل حل است:



یک مثال دیگر از حالتی که در نظر گرفتن کلمات میتواند برخی خطا ها را کاهش دهد به شرح زیر است:

DATA ON RULES & VERBS

	come	take	think	want
VP → V	9.5%	2.6%	4.6%	5.7%
VP → V NP	1.1%	32.1%	0.2%	13.9%
VP → V PP	34.5%	3.1%	7.1%	0.3%
VP → V SBAR	6.6%	0.3%	73.0%	0.2%
VP → V S	2.2%	1.3%	4.8%	70.8%

میبینیم که احتمال بیشینه برای هر قانون بسته به کلمه فعل فرق میکند و همین یعنی استفاده از کلمه به کمک ما آمده و برخی خطا ها را پوشش میدهد.

۳) این مورد به دلیل داشتن ذات احتمالاتی برای این حالت هم بر قرار است.
۴) این مشکل بدین صورت به طور حدی قابل حل است که با در نظر گرفتن کلمات و یا اطلاعات دیگر نظیر برچسب والد ها قوانین ما همپوشانی کمتر دارند و احتمال اینکه چنین حالتی برای ما رخ دهد کمتر است (چون قوانین متفاوت شده اند پس ترتیب رخ دادن آنها همواره منجر به ایجاد درخت های متفاوت و صحیح منطقی نخواهد شد) و بنابراین، این نوع خطا کمتر رخ خواهد داد.

منابع:

http://www.cs.columbia.edu/~mcollins/cs4705-spring2020/slides/parsing2.2.pdf
http://www.ling.helsinki.fi/kit/2009k/clt233/docs/Dickinson-pcfg.pdf
https://gawron.sdsu.edu/compling/course_core/lectures/pcfg/prob_parse.htm
https://www.cs.bgu.ac.il/~elhadad/nlp16/nlp03-statistical.html

https://cs.pomona.edu/~kim/CSC181S08/lectures/Lec12/Lec12.pdf

---

سوال ۵)

بخش a)

اطلاعتی که با NER میتوان از یک دیتای پزشکی استخراج کرد میتواند شمال نام دارو ها، نام بیماری ها، نام ویروس ها – میکروب ها و دیگر موجودیت های مشابه، نام بیمارستان ها یا مراکز درمانی مهم برای هر بخش یا تخصص خاص و نام پزشکان معروف و متخصص در یک زمینه خاص باشند. البته هر نوع اطلاعاتی که ماهیت نام و مکان و سازمان و ... داشته باشند قابل استخراج است اما مهم ترین آن ها در پاسخ تمرین سعی شد که گنجانده شود.

بخش b)

یک مدل ابتدایی برای این تسک طراحی میکنیم.

طراحی سیستم شامل دو بخش اصلی است.

آموزش و ارزیابی

۱) برای آموزش سیستم در ابتدا نیاز به داده داریم. داده باید برچسب دار باشد. از انکودینگ IO به دلیل راحت تر بودن و سریع بودن و اینکه در عمل بهتر از IOB است استفاده میکنیم. پس از آنکه داده ها را برچسب زنی کردیم (پر واضح است که بخش پیش پردازش داده بخش جدایی ناپذیر هر تسک NLP است و در اینجا اشاره نشد) باید اقدام به استخراج ویژگی ها کنیم.

میتوان از POS کلمه های قبل و بعد – خود کلمه های قبل و بعد و tag NER کلمه های قبلی به عنوان یک سری فیچر استفاده کرد. علاوه بر این برخی فیچر های دست نوشت نظیر مواردی که در زیر اشاره میشود میتوانیم داشته باشیم:

داشتن oxa در کلمه

داشتن : در کلمه

داشتن field در انتهای کلمه

و....

میتوان از فرم کلی کلمات نظیر داشتن اعداد – داشتن خط تیره – کوچک یا بزرگ بودن آنها یک shape خاص برای کلمات استخراج کرد و تحت این قوانین فیچر ها را بررسی کرد. مثلا کلمه ی CPA1 به صورت XXXd خواهد شد که X بیانگر حرف بزرگ (x هم بیانگر حرف کوچک) و d بیانگر عدد است. در صورتی که طول کلمه بیشتر از ۴ حرف باشد دو حرف اول و آخر را به این فرم در آورده و حرفهای میانی را صرفا به صورت

فرم در آورده و فقط نوع های موجود را به صورت canonical مرتب کرده و در نهایت میاوریم. مثال برای کلمه طولانی:

Varicella-zoster

دو حرف اول : Xx

دو حرف آخر: xx

حرفهای میانی شامل - x هستند که اگر مرتب شوند به صورت x- خواهند شد و در نهایت shape ما به صورت زیر خواهد شد که از چسباندن سه بخش بالا به دست می‌آید:

Xx-xxx

یک نمونه مجموعه فیچر به صورت زیر است:

Features

W_0	22.6
W_{+1}	%
W_{-1}	fell
T_{-1}	VBD
T_{-1}-T_{-2}	NNP-VBD
hasDigit?	true
…	…

که کلمه فعلی با W_0 و کلمه های بعدی و قبلی با اندیس مثبت و منفی – تگ کلمه ها با برچسب T و اندیس متناظر در جدول آمده است و بقیه فیچر ها هم به ترتیب آورده میشوند. نظیر داشتن عدد (hasDigit?) و …

این فیچر ها در نهایت در MEMM یا به طور دقیق تر در Maximum Entropy Markov Models استفاده خواهند شد.

MEMM ها برای مدل هایی به کار میرود که با Sequence سر و کار داریم. در این تسک هم چون داده ما به صورت Sequence و دنباله است.

میتوان از سه نوع آموزش Greedy – Beam – Viterbi برای مدل مارکوفی ما استفاده کرد. Viterbi از بین این نوع الگوریتم ها پیشنهاد میشود چون بهترین دنباله سراسری را انتخاب میکند.

همچنین میتوان از CRF یا Conditional Random Fields به عنوان یک مدل دنباله ای دیگر (غیر از MEMM) برای مساله استفاده کرد.

همچنین توضیحات دقیق تر برای CRF در لینک زیر موجود است:
https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776
که جزییات هر کدام از مدل ها در اسلاید ها آورده شده است و به دلیل خلاصه تر شدن توضیحات به آنجا ارجاع داده میشود.

برای استفاده از مدل های پیچیده در نظیر یادگیری عمیق میتوان از Bi-directional LSTM به همراه انکودینگ کلمات BERT یا GloVe استفاده کرد. که با CRF در نهایت برای ما خروجی را تولید خواهد کرد.
بسته به حجم داده و دقت مورد نیاز میتوان بین یک روش سنتی و یا روش مبتنی بر یادگیری عمیق یکی را انتخاب کرد.

پس از آموزش مدل حال باید به ارزیابی آن بپردازیم.
۲) ارزیابی Stanford برای NER بر اساس توکن ها نیست و بر اساس خود نوع موجودیت است. یعنی یک موجودیت ممکن است از چند توکن تشکیل شده باشد که باید تمامی آنها را در نظر گرفت.
میتوانیم از روش معمول Precision / Recall / F1 برای ارزیابی استفاده کنیم که پس از اینکه کلمات را برای تگ کردن به مدل آموزش داده شده دادیم، خروجی حاصل را با خروجی اصلی برای هر نوع موجودیت ممکن مقایسه کرده و بسته به اینکه تگ ها را چگونه پیش بینی کرده است جدول یا ماتریکس confusion را تشکیل میدهیم و از روی آن P, R, F1 را محاسبه میکنیم.
مشاهده میشود که در این نوع ارزیبای حتی اگر یک توکن از یک نوع موجودیت بزرگ اشتباه برچسب زنی شود معیار ارزیابی آن را اشتباه تشخیص داده و خطا در نظر گرفته میشود. میتوان از معیار های دیگر نظیر MUC که به صورت زیر بخش و بخشی امتیاز دهی میکنند استفاده کرد که با یک خطای کوچک بقیه تشخیص های درست سوخته نشوند.
معیار پیشنهادی ما MUC است.

منبع:
اسلاید ها و ویدیو های دانشگاه استنفورد – آقای Manning
https://www.aitimejournal.com/@akshay.chavan/complete-tutorial-on-named-entity-recognition-ner-using-python-and-keras