



گزارش پروژه پایانی - فاز نهایی

استخراج رابطه در زبان فارسی با بررسی وابستگی‌های جهانی Seraji و PerDT

مبانی پردازش زبان و گفتار

استاد: دکتر بهروز مینایی بیدگلی

اعضای گروه:

دانیال بازمانده ۹۷۵۲۱۱۳۵

محمدحسین کریمیان ۹۷۵۲۱۴۶۸

۱	هدف پروژه ۴
۲	مفهوم استخراج رابطه ۴
۳	وابستگی‌های جهانی ۵
۳.۱	وابستگی جهانی <i>Seraji</i> ۷
۳.۲	وابستگی جهانی <i>PerDT</i> ۸
۳.۳	مقایسه‌ی وابستگی‌های جهانی ۹
۴	فریم‌ورک <i>Predpatt</i> ۱۰
۵	آماده‌سازی دیتاست ۱۲
۵.۱	مرحله اول: جمع‌آوری داده‌ها ۱۲
۵.۲	مرحله دوم: پاکسازی داده‌ها ۱۵
۵.۳	مرحله سوم: پردازش آماری داده‌ها ۱۵
۶	پیاده‌سازی فرایند استخراج رابطه ۱۷
6.1	استخراج آرگومان‌ها با استفاده از وابستگی جهانی <i>Seraji</i> ۲۰
6.2	استخراج آرگومان‌ها با استفاده از وابستگی جهانی <i>PerDT</i> ۲۲
6.3	به دست آوردن روابط و اجرای وابستگی <i>Seraji</i> روی تمامی جملات ۲۳
7	<i>BERT</i> ۲۷
۷.۱	<i>ParseBERT</i> چیست؟ ۲۷
7.2	استفاده از <i>ParseBERT</i> در زبان فارسی ۲۸
۷.۳	روش استفاده ۲۸

۲۹.....	جمع آوری داده:	۷.۳.۱	
۲۹.....	پیش پردازش داده:	۷.۳.۲	
۲۹.....	بخش بندی متن:	۷.۳.۳	
۳۰.....	<i>Pre-training</i>	۷.۳.۴	
۳۰.....	مدل سازی نهایی	۷.۳.۵	
۳۰.....	پیاده سازی	7.4	
۳۲.....	<i>Training</i>		۸

۱ هدف پروژه

هدف اصلی و غایی این پروژه، پیاده‌سازی یک سیستم هوشمند با استفاده از ابزارهای پردازش زبان‌های طبیعی و یادگیری ماشین، در جهت استخراج رابطه از جملات زبان فارسی است. موضوع پروژه انتخابی در واقع، ترکیبی از دو پروژه از لیست پروژه‌های داده شده است. چرا که در این پروژه، استخراج رابطه با استفاده از هر دو وابستگی جهانی PerDT و Seraji انجام شده است.

۲ مفهوم استخراج رابطه

در ابتدا، به توضیح و تفصیل مفهوم استخراج رابطه (Relation Extraction) در پردازش زبان‌های طبیعی می‌پردازیم.

عملیات استخراج رابطه در آموزش مدل‌های هوش مصنوعی مبتنی بر پردازش زبان طبیعی برای یادگیری روابط موجودیت‌های مختلف در یک متن برای تجزیه و تحلیل داده‌های استخراج‌شده انجام می‌شود. تکنیک‌های مختلفی برای انجام استخراج موجودیت وجود دارد، از تطبیق رشته ساده تا رویکردهای فودکار پیچیده‌تر.

منظور از استخراج رابطه مجموعه تسک‌هایی است که به استخراج روابط معنایی موجود در متون می‌پردازد. این روابط معمولاً بین دو یا چند نهاد از یک نوع خاص (مانند شخص، سازمان، مکان و...) رخ می‌دهند و در تعدادی از دسته‌بندی‌های معنایی قرار می‌گیرند. (مانند متاهل، مجرد، استفاده‌شده و...) به عبارت بهتر، استخراج رابطه وظیفه‌ی پیش‌بینی صفات و روابط را برای موجودات و انتیتی‌های درون یک جمله دارد و مولفه‌ای کلیدی برای سافت گراف‌های دانش است.

به عنوان نمونه، در زبان انگلیسی جمله‌ی «Barack Obama was born in Honolulu, Hawaii» را در نظر بگیرید. در این جمله هدف پیدا کردن رابطه‌ی «bornInCity» است. این رابطه بین دو موجودیت (Entity) «Barak Obama» و «Honolulu» وجود دارد.

البته در این پروژه ما با جملات و دادگان فارسی کار خواهیم داشت و به پیاده‌سازی استخراج رابطه در زبان فارسی پرداخته‌ایم.

به عنوان نمونه در زبان فارسی، جمله‌ی «دانیال بازمانده در شهر مشهد به دنیا آمد.» را در نظر بگیرید. هدف از تست استخراج رابطه پیدا کردن رابطه‌ی «به دنیا آمدن» است. ما انتظار داریم پس از انجام موفق عملیات، آرگومان‌های این رابطه به دست آیند. در واقع، «دانیال بازمانده» و «شهر مشهد» آرگومان‌ها و «به دنیا آمد.» رابطه‌ای است که بین این دو آرگومان وجود دارد.

به عنوان نمونه‌ی دیگر می‌توان جمله‌ی «پاریس در فرانسه است.» را بیان کرد. در این جمله آرگومان‌ها «پاریس» و «فرانسه» هستند و «در» رابطه‌ای است که بین این دو موجودیت وجود دارد. همانطور که واضح است، الزامی به فعل بودن و معنی فعل دادن برای رابطه وجود ندارد.

انواع روش‌های مختلفی برای انجام عملیات استخراج رابطه وجود دارد:

- ۱) Rule-based RE
- ۲) Weakly Supervised RE
- ۳) Supervised RE
- ۴) Distantly Supervised RE
- ۵) Unsupervised RE

هر کدام از روش‌های فوق، مزایا و معایبی دارند که تفصیل آنها از موصله‌ی این گزارش خارج است.

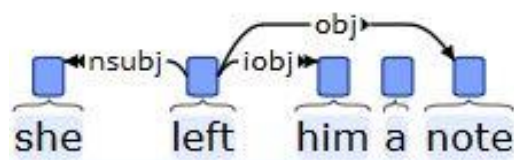
از کاربردهای مهم استخراج رابطه می‌توان به جستجوی سافتار یافته، تجزیه و تحلیل احساسات (Sentiment Analysis)، سیستم‌های پرسش و پاسخ (Question Answering) و خلاصه‌سازی متن می‌توان نام برد.

۳ وابستگی‌های جهانی

وابستگی جهانی (UD) پروژه‌ای است که در حال توسعه‌ی ماشینی‌نویسی از نظر زبانی سازگار برای بسیاری از زبان‌ها، با هدف تسهیل توسعه‌ی تجزیه‌کننده چندزبانه، یادگیری بین زبانی و تجزیه و تحلیل تمقیقات از منظر گونه‌شناسی زبان است. فلسفه‌ی کلی این است که فهرستی جهانی از مقوله‌ها و دستورالعمل‌ها برای تسهیل ماشینی‌نویسی ثابت سافتارهای مشابه در بین زبان‌ها فراهم کند.

به عبارت بهتر، وابستگی‌های جهانی (UD) چارچوبی برای ماشیه‌نویسی ثابت دستور زبان (بفش‌هایی از گفتار، ویژگی‌های صرفی و وابستگی‌های نحوی) در زبان‌های مختلف انسانی است. UD یک تلاش با بیش از ۳۰۰ مشارکت‌کننده است که نزدیک به ۲۰۰ درخت را از بیش از ۱۰۰ زبان تولید می‌کنند.

این طرح، تجزیه و تحلیل نحوی جملات را از نظر وابستگی‌های دستور زبان مربوطه تولید می‌کند. هر وابستگی از یک تابع نحوی مشخص می‌شود که روی لبه‌ی وابستگی (فلش در شکل) نشان داده می‌شود.



برای موفقیت یک UD فوب، موارد زیر باید رعایت شوند:

- در زمینه‌ی تحلیل زبانی برای هر زبان رضایت‌بفش و مناسب باشد.
- برای گونه‌شناسی زبانی فوب باشد. به‌عنوان مثال، زمینه‌ای مناسب برای ایجاد موازی زبانی بین زبان‌ها و خانواده‌های زبانی فراهم کند.
- برای ماشیه‌نویسی سریع و ثابت توسط یک ماشیه‌نویس انسانی مناسب باشد.
- به راحتی توسط یک غیر زبان‌شناس قابل درک و استفاده باشد.
- برای تجزیه کامپیوتری با دقت بالا مناسب باشد.
- به فوبی از وظایف درک زبان پایین دستی پشتیبانی کند. (استفراخ رابطه، درک مطلب، ترجمه ماشینی و...)

در زبان فارسی، وابستگی جهانی حاوی داده‌هایی از چندین بانک درختی (Treebank) است که توسط تیم‌های مختلف در زمان‌های مختلف و با ابزارهای تبدیل متفاوت ایجاد شده‌اند. در حال حاضر دو وابستگی جهانی مهم در زبان فارسی وجود دارند: Seraji و PerDT.

بانک درختی وابستگی جهانی فارسی (Persian UD) نسخه تبدیل‌شده‌ی بانک درختی وابستگی فارسی اوپسالا (UPDT) است. (سراجی، ۲۰۱۵). این بانک درختی، طرح ماشیه‌نویسی اصلی خود را بر اساس وابستگی‌های تایپ‌شده‌ی استنفورد دارد. (دی مارنر و همکاران، ۲۰۰۶؛ د مارنر و میننگ، ۲۰۰۸)

این طرح برای زبان فارسی گسترش یافت تا روابط نموی خاص زبان را در بر گیرد که نمی‌توانست توسط طرح اولیه توسعه‌یافته برای انگلیسی پوشش داده شود. این بانک درختی شامل ۶۰۰۰ جمله‌ی متون مکتوب با تنوع دامنه بزرگ، از نظر ژانرهای مختلف (شامل مقالات روزنامه، داستان‌ها، توضیحات فنی و اسناد در مورد فرهنگ و هنر) و نشانه‌گذاری است. تخییرات در نشانه‌گذاری به دلیل تخییرات املائی کلمات مرکب و عبارات ثابت در زبان است.

UPDT اصلی توسط مژگان سراجی و زیر نظر جواکیم نیور و کارینا جهانی در دانشگاه اوپسالا ساخته شد. جدای از طرح ماشیه‌نویسی جهانی و قوانین کلی در UD، UD فارسی و UPDT در نشانه‌گذاری تفاوت بیشتری دارند. تمام کلمات ماوی کلیده‌ای تقسیم‌نشده (قطعات ضمائر و کوپولا) که با برچسب‌های پیچیده در UPDT ماشیه‌نویسی شده‌اند، از کلیدواژه‌ها جدا شده‌اند و برچسب‌های متمایز در UD فارسی دریافت کرده‌اند.

تبدیل UPDT به وابستگی‌های جهانی به صورت نیمه خودکار انجام شده است. در این فرآیند، از یک اسکریپت تبدیل برای معکوس کردن سر و روابط وابسته در تعدیل‌کننده مرف اضافه (prep) و مفعول یک مرف اضافه (pobj) استفاده کردیم. علاوه بر این، ما از اسکریپت‌های دیگری که برای فارسی طراحی شده‌اند، برای جدا کردن انواع کلیتیک‌ها از میزبان خود استفاده کرده‌ایم. متعاقباً قوانین مختلفی را برای بازنویسی تگ‌های درشت قسمت گفتار و برچسب‌های وابستگی اضافه کردیم.

بانک درفتی وابستگی جهانی فارسی (PerUDT) حاصل پوشش خودکار بانک درفت وابستگی فارسی (PerDT) با اصلاحات دستی گسترده است که توسط محمدصادق رسولی، پگاه صفری، امیرسعید مولودی و علیرضا نوریان در سال ۲۰۱۳ پیاده‌سازی شده است.

بانک درفتی اصلی شامل ۲۹ هزار جمله است که از متون فارسی معاصر در ژانرهای مختلف از جمله: اخبار، مقالات دانشگاهی، مقالات مجلات و داستان‌ها نمونه برداری شده است. این بانک درفتی بر اساس یک طرح زبان خاص ماشیه‌نویسی شد و تبدیل خودکار آن شامل سه مرحله اصلی بود:

۱) بازبینی توکن‌سازی (Revising Tokenization)

۲) نگاشت POS (POS Mapping)

۳) نگاشت وابستگی (Dependency Mapping)

در مرحله‌ی نشانه‌گذاری، به منظور جداسازی عطف‌های چند کلمه‌ای افعال ساده که به صورت یک نشانه در PerDT گروه بندی شده اند، از دستورالعمل‌های موجود برای یافتن خودکار افعال اصلی پیروی کردیم. همچنین به طور خودکار کلیده‌های ضمیری را جدا کردیم.

در مرحله تبدیل *POS*، ما از آخرین تگر فارسی NER مبتنی بر (BERT (Taher et al., 2020) با اصلاحات دستی برای گسترش فراخوان استفاده کردیم. از طریق هفت موجودیت مختلف شناسایی‌شده توسط برچسب‌گذار، از Person و Location برای علامت‌گذاری برچسب‌های PROPEN استفاده کردیم. در مجموع، PerDT شامل ۴۳ رابطه‌ی نمودی بدون نگاشت مستقیم است. مروف ربط از ابتدای جمله تا انتها مرتب شده اند و مهمتر از آن، مروف اضافه به عنوان سر عبارات اضافه و افعال کمکی به عنوان سر جملات قرار می‌گیرند.

بنابراین ما ترتیب مروف ربط را از انتها به ابتدا از طریق یک اسکریپت و قواعد مناسب برای تبدیل هر نوع رابطه به نسخه UD آن به درستی مرتب کردیم. در طول کل فرآیند و در پایان هر مرحله، نتایج را بررسی کردیم و در صورت نیاز اصلاحات دستی را اعمال کردیم.

۳/۳ مقایسه‌ی وابستگی‌های جهانی

در تصویر زیر، جدولی از مقایسهٔ «درخت بانک وابستگی فارسی اوپسالا» با هریک از وابستگی‌های جهانی Seraji و PerDT ارائه داده است. مینا را بر وابستگی‌های جهانی می‌گذاریم و نشان می‌دهد هر برپسب اجزای سفن در وابستگی جهانی چه معادلی در پروژه‌ی دانشگاه اوپسالا (Seraji) و «پروژه‌ی دادگان وابستگی زبان فارسی (PerDT)» دارد.

Dad Sub-Tag Fa	Dad Sub-Tag	Dad Fa Name	Dad Tag	Seraji Fa Name	Seraji Name	Seraji Tag	Persian Name	UD Name	UD Tag	UD Num
صفت مطلق	AJP			صفت	Adjective	ADJ				
صفت عالی	AJSUP			صفت عالی	Superlative adjective	ADJ_SUP				
صفت تفضیلی	AJCM	صفت	ADJ				صفت	adjective	ADJ	1
صفت شمارشی	POSNUM			صفت معمولی	Participle adjective	ADJ_INO				
				صفت ندا	Vocative adjective	ADJ_SUP				
				صفت مقایسه‌ای	Comparative adjective	ADJ_CMPR				
				حرف اضافه پیشین	Preposition	P	حرف اضافه	adposition	ADP	2
				قید	Adverb	ADV				
				قید مقایسه	Adverb of Comparison	ADV_COMP				
				قید استهلام	Adverb of Interrogation	ADV_I				
				قید مکان	Adverb of Location	ADV_LOC				
				قید نفی	Adverb of Negation	ADV_NES				
				قید زمان	Adverb of time	ADV_TIME				
		شبه جمله	PSUS							
		جزء دستوری	PART							
وچهی	MODL	فعل	V	فعل کمکی	Auxiliary Verb	V_AUX	فعل کمکی	auxiliary	AUX	4
		نقش نمای همپایگی	CONJ	حرف ربط	Conjunction	CON	حرف ربط هم پایه ساز	coordinating conjunction	CCONJ	5
صفت تعجبی	EXAJ	پیش توصیف گر	PREM	مشخصگر	Determiner	DET	مشخصگر	determiner	DET	6
صفت پرسشی	QUAJ									
صفت اشاره	DEMAJ									
صفت مبهم	AMBAJ									
نقش نمای ندا پیشین	PRADR	نقش نمای ندا	ADR	صوت و حرف ندا	Interjection	INT	صوت و حرف ندا	interjection	INTJ	7
نقش نمای ندا پسین	POSADR									
چاندان	ANM	شبه جمله	PSUS							
بی جان	IANM	اسم	N							
		شاخص	IDEN							
		صفت شمارشی پیشین	PRENUM							
		صفت شمارشی پسین	POSNUM							
		حرف اضافه پسین	POSTP	علامت معمولی	CLUTIC		ادات	particle	PART	10
ضمیر شخصی جدا	SEPER									
ضمیر شخصی پیوسته	JOPER									
ضمیر اشاره	DEMON									
ضمیر پرسشی	INTG									
ضمیر بارزانی مشترک	CREFX									
ضمیر بارزانی غیرمشترک	UCREFX									
ضمیر متقابل	RECPR									
		ندارد					اسم خاص	proper noun	PROPN	12
		علامت نگارشی وابستگی	PUNC	جداکننده	DELIM		علامت نگارشی	punctuation	PUNCT	13
		نقش نمای وابستگی	SUBR	حرف ربط	Conjunction	CON	حرف ربط وابسته‌ساز	subordinating conjunction	SCONJ	14
		ندارد								
				نماد	Symbol	SYM	نماد	symbol	SYM	15
				فعل امری	Imperative verb	V_IMP				
				فعل زمان گذشته	Past tense verb	V_PA				
				فعل گذشته کامل	Past participle verb	V_PP				
				فعل ربطی	Verb copula	V_COP				
				فعل زمان حال	Present tense verb	V_PRS				
				فعل شرطی (انتزاعی)	Subjunctive verb	V_SUB				
معلوم	ACT									
مجهول	PASS									
وچهی	MODL									
		شبه جمله	PSUS	واژه خارجی	Foreign word	FW	دیگر واژه ها	other	X	17

Predpatt مخفف عبارت Predicate-Argument Extraction from Universal Dependencies

یک کتابخانه‌ی قدرتمند توسعه‌یافته است که به عنوان چارچوبی از الگوهای استخراج گزاره (Predicate) - آرگومان قابل توسعه، قابل تفسیر و زبان فنتی ارائه می‌شود. PredPatt عملیات تجزیه نمودی عمیق پروژه‌ی وابستگی جهانی را به یک لایه معنایی کم‌عمق اولیه پیوند می‌دهد. این می‌تواند پایه‌ای برای لایه‌بندی آینده‌ای از ماشین‌نویسی‌های معنایی در بالای درخت‌های وابستگی جهانی باشد. همچنین به‌طور جداگانه می‌تواند به عنوان یک مؤلفه زبان‌شناختی به‌فوبی از سازوکار «IE Universal» در نظر گرفته شود.

این کتابخانه در سراسر زبان‌های دنیا کار می‌کند. *Predpatt* با سایر سیستم‌های برپشته‌ی Open IE مقایسه شده است و نتایج به‌دست آمده نشان می‌دهد که PredPatt به بهترین دقت و یادآوری دست می‌یابد.

کاربرد اصلی *PredPatt* استخراج روابط از جملات ورودی به آن است. به این صورت که تلاش می‌کند با استفاده از الگوریتم‌های قوی پیاده‌سازی‌شده، آرگومان‌ها و رابطه‌های موجود در یک جمله را به دست آورد. با استفاده از این آرگومان‌ها و رابطه‌ها می‌توان گراف معنایی (*Semantic Graph*) مرتبط با جملات را ترسیم کرد.

به عنوان نمونه می‌توان کاربرد آن را در زبان انگلیسی مشاهده کرد. همانطور که مشخص است، آرگومان‌های *a* و *b* از جمله به دست آمده است که هرکدام از آرگومان‌ها نقش *POS* مخصوص به خود را دارند.

?a extracts ?b from ?c

?a: PredPatt

?b: predicates

?c: text

?a gave ?b to ?c

?a: Chris

?b: the book

?c: Pat

به عنوان نمونه‌ی دیگر در جملات زبان فارسی داریم:

علی دایی بازیکن فوتبال بازنشسته تیم ملی ایران و باشگاه پرسپولیس است.

a ? بازیکن b ? است.

a ? علی دایی

b ? فوتبال (بازنشسته تیم ملی ایران و باشگاه پرسپولیس)

در مثال بالا، a و b آرگومان‌های ما هستند. یعنی در این جمله، علی دایی و فوتبال آرگومان‌های رابطه ممسوب می‌شوند. همچنین کلمه‌ی «بازیکن» به عنوان رابطه‌ی بین این دو آرگومان ممسوب می‌شود.

نکته مهم: برای تشخیص اینکه کدام کلمه به عنوان رابطه در نظر گرفته شود، ما همیشه تمام کلماتی که بین دو آرگومان می‌آیند را به عنوان رابطه‌ی آن دو آرگومان در نظر می‌گیریم. برای استناد به درستی این کار، از مقاله‌ی Farsbase bkp نوشته دکتر مینایی بهره گرفته‌ایم.

سهراب سپهری شاعر، نویسنده و نقاش اهل ایران بود.

a ? شاعر بود

a ? سهراب سپهری

a ? نویسنده

a ? سهراب سپهری

a ? نقاش b ?

a ? سهراب سپهری

b ? اهل ایران

مطابق توضیحات قبل، دو آرگومان «سهراب سپهری» و «اهل ایران» داریم و کلمات «شاعر»، «نویسنده» و «نقاش» نیز به عنوان روابط بین این آرگومان‌ها در نظر گرفته می‌شوند. پس سه رابطه در مجموع داریم.

۵ آمادہ سازی دیتا ست

۵/۱ مرحله اول: جمع آوری داده‌ها

به عنوان اولین فاز، امتیاج به جمع‌آوری مقدار زیادی داده داریم. زیرا برای دستیابی به هدف نهایی، نیاز به تعداد جملات زیادی داریم تا عملیات آموزش و آزمون را روی آنها انجام دهیم و مدل خود را بهبود دهیم. به عنوان سورس و مرجع از دیتاست سایت جامع ویکی‌پدیا استفاده می‌کنیم.

برای دستیابی به داده‌ها و جملات موجود در این وبسایت بزرگ، به `Crawl` کردن آنها می‌پردازیم. برای این کار، با استفاده از کتابخانه‌ی `BeautifulSoup` اقدام به خزش در فایل‌های `html` می‌کنیم و اطلاعات را مرحله به مرحله استخراج می‌کنیم. توضیحات مربوط به این بخش به طور کامل در گزارش فاز قبلی آمده است و اینجا به طور فاصله به مراحل انجام کار اشاره می‌کنیم و به همین بسنده می‌کنیم.

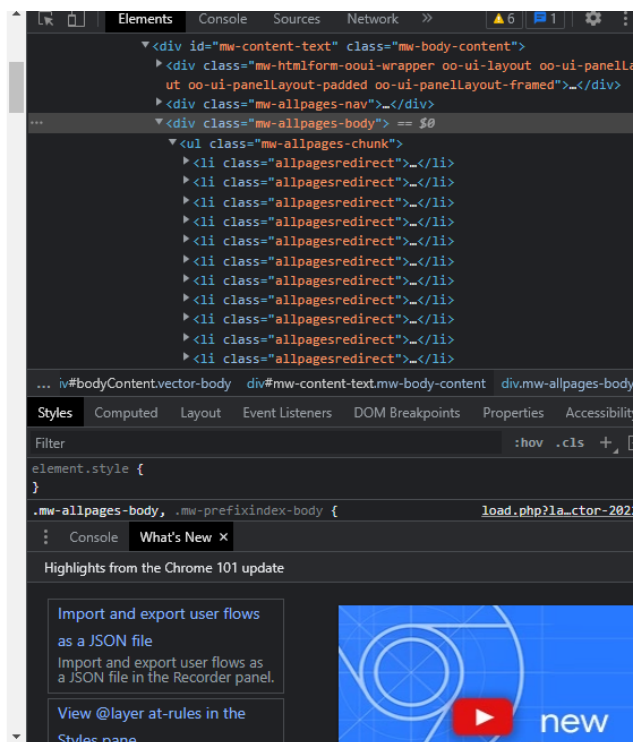
در گام اول، ابتدا تمامی الفباهای موجود در این وبسایت را به دست می‌آوریم:



در گام بعد، اقدام به دستیابی به تمام موضوعاتی که با آن مروف شروع می‌شوند، می‌کنیم. به این خاطر که داده‌های مورد نیاز ما حداکثر سی هزار جمله می باشد، فقط موضوعاتی که در صفحه اول مروف هستند را در نظر می‌گیریم و همچنین مروفی از فهرست که بیش از یک مرف دارند را هم حذف می‌کنیم.

صفحة قبل (Zevtera) | صفحه بعد (اتومیلیستا)

ا
• ا-10
• ا-10 تاندربولت
• ا-10 تاندربولت 2
• ا-4 اسکای هاوک
• ا-4 اسکای هاوک
• ا-4 اسکای هاوک
• ا-6 اینترودر
• ا-6 اینترودر
• ا-7 کرسر
• ا-7 کرسر
• ا-86929
• ا-سانتالول
• ا-تلکس
• ا-تلکس
• ا-تلکس
• ا-تلکس
• ا-می
• ا-می
• ا-نه-ب
• ا-ها
• ا-100
• ا-100 تاندربولت
• ا-100 تاندربولت 2
• ا-4 اسکای هاوک
• ا-4 اسکای هاوک
• ا-4 اسکای هاوک



```

https://fa.wikipedia.org/wiki/آ
https://fa.wikipedia.org/wiki/آ_آلبر
https://fa.wikipedia.org/wiki/آ_اس_رم
https://fa.wikipedia.org/wiki/آ_اس_پ.
https://fa.wikipedia.org/wiki/آ_ب_ث_آفریقا
https://fa.wikipedia.org/wiki/آ_ب_ث_آفریقا
https://fa.wikipedia.org/wiki/آ_ب_س_آفریقا
https://fa.wikipedia.org/wiki/آ_ب_س_آفریقا
https://fa.wikipedia.org/wiki/آ_ث_میلان
https://fa.wikipedia.org/wiki/آ_ث_میلان
https://fa.wikipedia.org/wiki/آ_ج_آزیری
https://fa.wikipedia.org/wiki/آ_دوبو
https://fa.wikipedia.org/wiki/آ_والتون_لیتز
https://fa.wikipedia.org/wiki/آ_والتون_لیتز
https://fa.wikipedia.org/wiki/آ_کوتنابالی
https://fa.wikipedia.org/wiki/آ_کیتمن_هو
https://fa.wikipedia.org/wiki/آ_گ.
https://fa.wikipedia.org/wiki/آ_ار.د
https://fa.wikipedia.org/wiki/آ_اس_رم
https://fa.wikipedia.org/wiki/آ_اس.
https://fa.wikipedia.org/wiki/آ_اس_ترگو_مورش
https://fa.wikipedia.org/wiki/آ_اس_پ.
https://fa.wikipedia.org/wiki/آ_اس_رم.
https://fa.wikipedia.org/wiki/آ_اس_مونکو
https://fa.wikipedia.org/wiki/آ_ث_میلان
https://fa.wikipedia.org/wiki/آ_ث_میلان_و_تیم_ملی_فوتبال_ایتالیا
https://fa.wikipedia.org/wiki/آ_ث_میلان

```

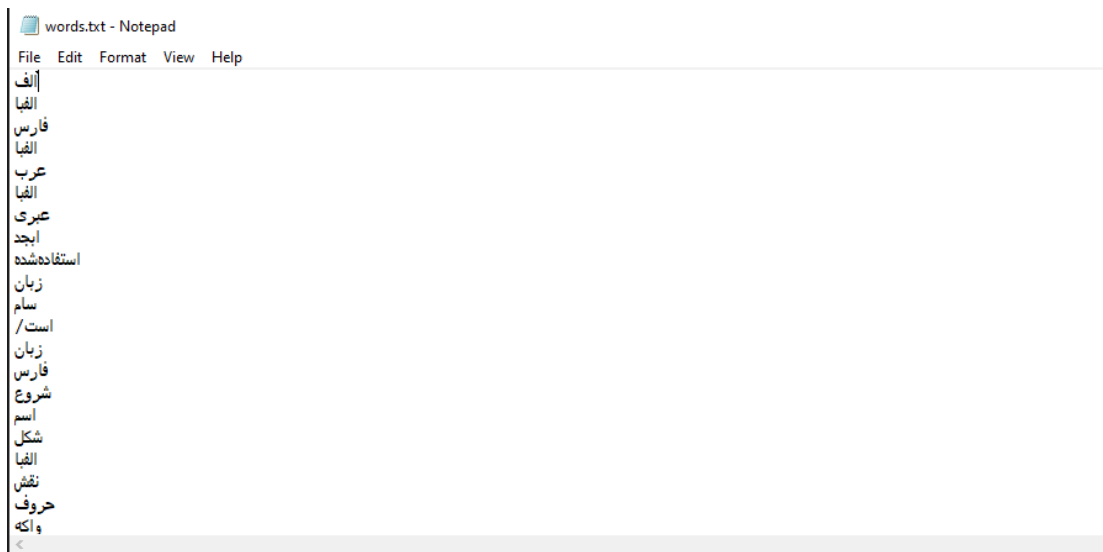
مال با ایجاد یک درخواست (*request*) به هر کدام از لینک‌های موجود، اقدام به دستیابی به جملات موجود در هر صفحه می‌کنیم. برای هر صفحه یک موجودیت *json* شامل *id*، *title* و *subject* درست می‌کنیم.

```

{id: 1,
"title": "الف",
"text": "اولین حرف الفبای فارسی و الفبای عربی و الفبای عبری و سایر ابجد های استفاده شده توسط زبان های سامی است. این «الف» در نقش یکی از حروف واکه بلند (صوت بلند) است که گاه «n» حرف در زبان فارسی شروع کننده خیلی از اسم هاست به صورت (ا) در کلماتی چون: آب، باد و دریا است. و هم در نقش تلفظ همزه به کمک حرف های واکه به کار می رود، مانند: آبر، انسان و استاد (با حروف واکه کوتاه فتحه - کسره و ضمه) و مانند: آب (الف اول در اصل=اب)، او، این (با حروف واکه بلند) آ یا الف قددار (به عربی: ألف ممدودة) است که آن را با همزه یا الف یکی «n» در لغتنامه ها «ا» مخفف اسم است «n» است این نشانه با نام های مختلفی «n» می شناسند، اما در حقیقت حرف جداگانه ای است. این حرف تغییر شکلیافته حرف عبری «א» است فرهنگ معین در آغاز فصل «آ، ا» نوشته شده است: «آ» و «ا» را در الفبای فارسی یک حرف به حساب «n» معرفی شده است، از جمله آورند، اما درحقیقت دو حرف جداگانه اند...» در لغتنامه دهخدا نیز آمده است: «آ. (حرف) الف لیثنه، مقابل همزه یا الف متحرکه، حرف اول است از حروف هجا و در حساب جُمْل آن را به «یک» دارند.» اما این نشانه درحقیقت ترکیبی از دو حرف از الفبای فارسی است: «ا» به عنوان صامت که شکلی از آشکال همزه است، و «آ» که مصوت بدون نامی است از مصوت های شُکَّانَه خط فارسی. اما به اشتباه در دستور خط فارسی مصوب فرهنگستان زبان و ادب فارسی، در «جدول ۱. نشانه های خط فارسی، نشانه های اصلی»، سی و سه نشانه معرفی شده است و این نشان «آ» جزو نشانه های سی و سه گانه ای که الفبای فارسی را تشکیل می دهند آورده نشده است. تمام مطالب این بخش از لغتنامه دهخدا، ماده «آ»، نقل شده است؛ مگر آن که از منبع دیگری نام برده شده باشد «n».
```

۵/۲ مرحله دوم: پاکسازی داده‌ها

در این مرحله با استفاده از کتابخانه‌ی *nlTK* در زبان انگلیسی است، اقدام به نرمال‌سازی هرکدام از جملات می‌کنیم. سپس برای به دست آوردن کلمات، با *stemming* ریشه اصلی آن‌ها را به دست می‌آوریم و در نهایت چک می‌کنیم که این کلمات بین *stop words* های زبان فارسی هستند یا خیر و اگر در فایل *stop words* نبود، آن را در لیست کلمات در فایل *words.txt* می‌نویسیم که در شکل زیر این فایل را می‌بینیم.



۵/۳ مرحله سوم: پردازش آماری داده‌ها

در نهایت تمامی جملات به دست آمده را جدا می‌کنیم و در فایل *sentences.json*، همراه با آیدی و موضوع آن می‌نویسیم. از این جملات به عنوان دادگان اصلی در جهت استفاده در بخش‌های بعدی استفاده می‌کنیم.

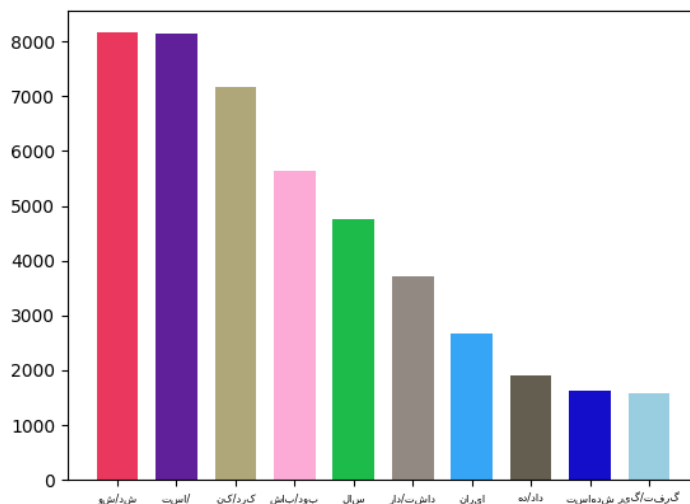
```
{
  "id": 30501,
  "text": "بی‌ها در رده «پیروان حزب» (پیروان) طبقه‌بندی شد (NSDAP) پدر هابرماس از سال ۱۹۳۳ عضو حزب ناسیونال سوسیالیست",
  "subject": "یورگن هابرماس"
},
{
  "id": 30502,
  "text": "ل تجویز می‌شد، عضوی از سازمان جوانان (جونگولک) شد که شاخه‌ای از سازمان شبه‌نظامی جوانان هیتلری بشمار می‌رفت",
  "subject": "یورگن هابرماس"
},
{
  "id": 30503,
  "text": "همچنان فرائر از محدودیت سنی در آن سازمان ماند و مجبور نشد در ۱۴ سالگی به سازمان جوانان هیتلری بپیوندد",
  "subject": "یورگن هابرماس"
},
{
  "id": 30504,
  "text": "ن نیز دوباره داوطلب شده بود، خود را تا زمانی که آمریکایی‌ها منطقه را اشغال کردند از افسران پلیس پنهان کرد",
  "subject": "یورگن هابرماس"
},
}
```

تعداد جملات نهایی ۳۱۳۶۳ عدد می‌باشند که برای استفاده در وابستگی‌های جهانی مورد استفاده خواهند بود.

همچنین برای این که هر کلمه را با تعداد تکرارش داشته باشیم، یک دیکشنری تعریف می‌کنیم و هر بار که کلمه‌ای تکرار شد یکی به تعداد تکرار آن اضافه می‌کنیم و در نهایت آن را پس از *sort* کردن بر اساس تکرار بیشتر، در فایل *count.txt* ذخیره می‌کنیم.

```
count.txt - Notepad
File Edit Format View Help
شد/شو : 8162
است : 8139
کرد/کن : 7173
بود/باش : 5650
سال : 4763
داشت/دار : 3722
ایران : 2665
داد/ده : 1914
شده/است : 1632
گرفت/گیر : 1587
شهر : 1194
هست : 1177
کار : 1149
کشور : 1121
انگلیسی : 1086
توانست/توان : 1052
تاریخ : 1042
گفت/گو : 1017
کتاب : 932
بن : 916
زمین : 882
دست : 807
یافت/یاب : 805
رفت/رو : 766
اهدأ : 766
جهان : 741
گروه : 734
شرکت : 722
نظر : 714
آلمان : 711
آب : 710
تولید : 690
مم‌توان : 670
دوران : 659
گشت/اگر د : 657
علی : 657
```

نمودار فراوانی کلمات در زیر قابل مشاهده است:



۶ پیاده‌سازی فرایند استخراج رابطه

در این مرحله، اقدام به پیاده‌سازی سیستمی در جهت استخراج روابط موجود در جملات می‌کنیم. به نحوی که سیستم مربوط با دریافت یک یا تعدادی از جملات و همچنین انتخاب نوع وابستگی جهانی (*Seraji* یا *PerDT*)، آرگومان‌های مربوطه و رابطه‌ی بین آرگومان‌ها را استخراج کند.

توجه: لازم به ذکر است در جهت توسعه‌ی یک سیستم نرم‌افزاری مطلوب در جهت راحتی مشتریان و *UI/UX* بهتر، اقدام به پیاده‌سازی این سیستم در قالب یک سیستم *client/server* کردیم و تلاش کردیم صفحه‌ای به عنوان رابط کاربری برای کاربر نمایش دهیم تا به واسطه‌ی *API* های سرور بتواند به استخراج آرگومان‌های مربوطه از جملات دلفواه خود بپردازد. پیاده‌سازی این بخش نرم‌افزاری جزو قسمت‌های پروژه نبوده است.

برای بخش سرور از فریم‌ورک *Django* مربوط به زبان پایتون استفاده کرده‌ایم. با استفاده از این فریم‌ورک، یک صفحه طراحی کرده‌ایم که کاربر بتواند با ورودی متن مربوطه و انتخاب نوع وابستگی جهانی و انتخاب گزینه *Submit*، نتیجه را مشاهده کند.

همچنین اقدام به پیاده‌سازی *API* های مربوطه برای هر دو وابستگی جهانی پرداختیم که بتوان از دیوایس‌ها و برنامه‌های دیگر نیز با استفاده از این *API* ها ارتباط برقرار کرد.

RelationExtractionInPersian

RelationExtractionInPersian is a relation extraction framework for Persian texts

Choose your desired Universal Dependency:

☒ Seraji ☐ PerDT

Enter your text in Persian:

علی دایی بازیکن فوتبال بارزشسته تیم ملی ایران و باشگاه پرسپولیس است.

SUBMIT

Conll-Format:

text = علی دایی بازیکن فوتبال بارزشسته تیم ملی ایران و باشگاه پرسپولیس است.

1	علی	علی	NOUN	N_SING	Number=Sing	3	nsubj	-	-
2	دایی	دایی	NOUN	N_SING	Number=Sing	1	flat	-	-
3	بازیکن	بازیکن	NOUN	N_SING	Number=Sing	0	root	-	-
4	فوتبال	فوتبال	NOUN	N_SING	Number=Sing	3	nmod:poss	-	-
5	بارزشسته	بارزشسته	ADJ	ADJ	Degree=Pos	4	amod	-	-
6	تیم	تیم	NOUN	N_SING	Number=Sing	4	amod:poss	-	-

Output:

a? بازیکن b? است
a?: علی دایی
b?: فوتبال بارزشسته تیم ملی ایران و باشگاه پرسپولیس

RelationExtractionInPersian

RelationExtractionInPersian is a relation extraction framework for Persian texts

Choose your desired Universal Dependency:

☐ Seraji ☒ PerDT

Enter your text in Persian:

علی دایی بازیکن فوتبال بازنشسته تیم ملی ایران و باشگاه پرسپولیس است.

SUBMIT

Conll-Format:

text = علی دایی بازیکن فوتبال بازنشسته تیم ملی ایران و باشگاه پرسپولیس است.

1	علی	PROPN	N_ANM	Number=Sing	3	nsubj	0	3
2	دایی	PROPN	N_ANM	Number=Sing	1	flat:name	4	8
3	بازیکن	NOUN	N_ANM	Number=Sing	0	root	9	15
4	فوتبال	NOUN	N_IANM	Number=Sing	3	nmod	16	22
5	بازنشسته	ADJ	ADJ_AJP	None	4	amod	23	31

Output:

a? بازیکن b? است
a?: علی دایی
b?: فوتبال بازنشسته تیم ملی ایران و باشگاه پرسپولیس

برای استخراج آرگومان‌های مربوطه به پیاده‌سازی هر دو وابستگی جهانی پرداختیم:

۶/۱ استخراج آرگومان‌ها با استفاده از وابستگی جهانی Seraji

در این قسمت، پس از دریافت *ud_type* و *text* از کاربر، در صورتی که کاربر وابستگی جهانی سرابی را انتخاب کرده باشد، اقدام به تولید یک فایل موقتی به نام *file.txt* می‌کنیم. تمامی جملات ورودی را در این فایل می‌ریزیم. سپس اقدام به اجرا کردن یک کامند از طریق *bash* می‌کنیم.

دلیل استفاده از این فایل موقتی به دلیل نیاز این کامند به خواندن اطلاعات از فایل دارد. (*data=@file.txt*) مقدار پارامتر *model* را برابر *persian* قرار دادیم تا از مدل‌های مربوط به زبان فارسی برای استخراج رابطه استفاده شود. از *normalized_space* به عنوان *tokenizer* استفاده می‌کنیم و همچنین برای تجزیه‌گر (پارسر) از *api* ای که برای *udpipe* تعبیه شده است، استفاده می‌کنیم.

در صورتی که اجرای کامند فوق دچار خطا نشود، از کلید *result* در نتایج این دستور می‌توانیم فرمت *conllu* موردنظر را به دست آوریم.

منظور از Conllu Format چیست؟

CoNLL نامی معمولی برای قالب‌های *TSV* (مقادیر جدا شده با تب مانند *CSV*) در *NLP* است و در واقع فرمتی خاص برای نمایش مقادیر است که در این موزه بسیار پر کاربرد است. بسیاری از فرمت‌های مختلف *CoNLL* وجود دارد زیرا *CoNLL* هر سال یک کار مشترک متفاوت است که در کنفرانس سالانه *NLP* تصمیم‌گیری می‌شود. فرمت مشهور آن به این صورت است که:

- هر توکن در یک خط نشان داده می‌شود.
- هر جمله با یک خط خالی (*NewLine*) از جمله بعدی جدا می‌شود.
- هر ستون نشان‌دهنده یک *Annotation* است.
- هر کلمه در یک جمله دارای تعداد ستون‌های یکسانی است. (در برخی قالب‌ها: هر کلمه در مجموعه تعداد ستون‌های یکسانی دارد.)

- و... که در این گزارش نمی‌گنجد.

در ادامه، با استفاده از تابع *load_conllu* اقدام به خواندن فرمت *Conll* تولید شده می‌کنیم و تمامی تگ‌های *POS* به‌دست آمده و اطلاعات لازم را در یک لیست نگه‌داری می‌کنیم.

به عنوان مرحله نهایی و در قدم آخر، با ورودی دادن این لیست به کلاس *PredPatt* روابط مورد نظر و درفت رابطه ایجاد می‌شود. برای اینکه آرگومان‌های مربوطه را ببینیم، متود *pprint()* را از این کلاس صدا می‌زنیم.

مواردی که در بالا توضیح داده شد، در تصویر زیر قابل مشاهده است:

```
if request.method == "POST":
    ud_type, text = request.POST.get("ud_type"), request.POST.get("input_text")

    if ud_type == 'seraji':
        with open('file.txt', 'w', encoding='utf-8') as f:
            f.write(text.replace('\r', '').strip())
            f.close()

        bash_command = """curl -F data=@file.txt -F model=persian -F tokenizer=normalized_spaces -F tagger= -F
        parser= http://lindat.mff.cuni.cz/services/udpipe/api/process """
        try:
            send_order = subprocess.check_output(bash_command.split(), shell=True)
        except subprocess.CalledProcessError as e:
            os.remove("file.txt")
            raise RuntimeError("command '{}' return with error (code {}): {}".format(e.cmd, e.returncode, e.output))

        os.remove("file.txt")
        conll = json.loads(send_order)['result']
        ppatt = PredPatt([ud_parse for sent_id, ud_parse in load_conllu(conll)][0])
        return render(request, 'service.html',
            {'input_text': text, 'ud_type': ud_type, 'conll_text': conll, 'output_text': ppatt.pprint()})
```

۶/۲ استخراج آرگومان‌ها با استفاده از وابستگی جهانی *PerDT*

پیاده‌سازی این قسمت در بعضی بخش‌ها مشابه وابستگی جهانی سراجی است. در ابتدا اقدام به دانلود مدل‌های شبکه‌های عصبی موازی برای زبان فارسی می‌کنیم. برای دانلود این مدل‌ها از پکیج *stanza* استفاده می‌کنیم. با استفاده از متود *Pipeline* اقدام به انتخاب شبکه عصبی پیش‌فرض در زبان فارسی می‌کنیم. حال که مدل مربوطه را به دست آوردیم، جمله ورودی را به عنوان ورودی به مدل می‌دهیم.

فروچی مدل تعدادی جملات برای ما فواید بود که این جملات شامل تعدادی کلمات هستند. در این مرحله بر خلاف روش سراجی که خودش همه نقش‌ها را به ما می‌داد، خودمان باید یکی یکی ویژگی‌های مورد نیاز را به ازای هر کلمه به دست آوریم. ویژگی‌هایی از جمله شناسه، متن کلمه، لم، جایگاه آن در متن، *xpos*، مرف شروع و پایانی و...

تمامی ویژگی‌های مورد نیاز را خودمان دستی تولید می‌کنیم و در قالب یک رشته به همدیگر می‌پسباییم تا مانند سراجی به *PredPatt* دهیم تا فروچی را برایمان تولید کند. در واقع، فرمت *conll* را در اینجا خودمان تولید کردیم و به *Predpatt* دادیم. بر خلاف روش سراجی که با ران کردن کامند مربوطه این کار را انجام می‌داد. مواردی که در بالا توضیح داده شد، در تصویر زیر قابل مشاهده است:

```
elif ud_type == "perdt":
    stanza.download('fa') # This downloads the Persian models for the neural pipeline
    nlp = stanza.Pipeline('fa') # This sets up a default neural pipeline in Persian
    doc = nlp(text)
    for sentence in doc.sentences:
        res = "# text = " + text.strip()
        res = res.strip() + '\n'
        for word in sentence.words:
            res = res + str(word.id) + "\t" + str(word.text) + "\t" + str(word.lemma) + "\t" + \
                str(word.pos) + "\t" + str(word.xpos) + "\t" + str(word.feats) + "\t" + \
                str(word.head) + "\t" + str(word.deprel) + "\t" + str(word.start_char) + "\t" + \
                str(word.end_char)
            res = res + '\n'

    ppatt = PredPatt([ud_parse for sent_id, ud_parse in load_conllu(res)][0])
    return render(request, 'service.html',
                  {'input_text': text, 'ud_type': ud_type, 'conll_text': res, 'output_text': ppatt.pprint()})
```

۶/۳ به دست آوردن روابط و اجرای وابستگی *Seraji* روی تمامی جملات

در این مرحله، از *API* هایی که در مرحله قبل برای استخراج آرگومان‌ها با استفاده از وابستگی جهانی سرابی پیاده‌سازی کردیم، استفاده می‌کنیم تا هر یک از آرگومان‌ها را در تمامی جملاتی که داریم، به همراه رابطه‌ی بین این آرگومان‌ها به دست آوریم. در نهایت، دیتاست محدودتری به دست می‌آوریم که شامل تمامی جملاتی است که حداقل یک رابطه در درون خود دارند. (ممکن است تعدادی از جملات طبق مدل *Seraji* اصلاً شامل رابطه نباشند.)

ابتدا اقدام به باز کردن فایل جملات و هم‌پنین فایل جملات بعلاوه روابط آنها (برای نوشتن) می‌کنیم. سپس، با استفاده از آدرسی که برای *API* مدل سرابی تعیین کردیم، در حالتی که سرور در حال اجرا باشد، اقدام به ریکوئست زدن می‌کنیم.

```
sentences_file = open('../data/sentence_broken/sentences.json', 'r', encoding='utf8')
results = open('../data/sentence_broken/sentences_with_relations.json', 'w', encoding='utf8')

seraji_request_url = 'http://127.0.0.1:8000/api/seraji'

sentences_json = json.load(sentences_file)

counter = 29000
subject_counter = 2159
error_counter = 0
dataset = []
```

در واقع، روی تک تک جملات دیتاست خود، آنها را به عنوان ورودی *input_text* به *API* مربوطه تمت ریکوئست می‌دهیم. سپس، از فروبی به دست آمده با استفاده از عملیات مربوط به کار با رشته‌ها در پایتون، ابتدا اطمینان پیدا می‌کنیم که متما دو آرگومان وجود داشته باشند. (در صورت وجود تک آرگومان، رابطه تعریف نمی‌شود.) سپس در صورت وجود دو آرگومان، چک می‌کنیم که *subject* صفحه ویکی‌پدیا در آرگومان اول جمله باشد و بالعکس. در صورت وجود، آرگومان دوم را *object* می‌نامیم و تمامی کلمات بین این دو آرگومان را به عنوان رابطه‌ی بین این دو آرگومان در نظر می‌گیریم. (پیش‌تر توضیح داده شد که با استناد به مقاله‌ی *Farsbase pbk* دکتر مینایی این فرض را کرده‌ایم.)

```

for sentence_dict in sentences_json[29000:]:
    try:
        response = requests.post(seraji_request_url, json={'input_text': sentence_dict['text']})
        output_text = response.json()['output_text']

        subj_index = output_text.find('?a:')
        obj_index = output_text.find('?b:')
        if subj_index != -1 and obj_index != -1:
            subject = output_text[subj_index + 3:obj_index].strip()
            if sentence_dict['subject'] in subject or subject in sentence_dict['subject']:
                object = output_text[obj_index + 3:output_text.find('\n', obj_index + 3)]
                if 'SOMETHING' not in object and len(subject) >= 2:
                    relation = output_text[output_text.find('?a') + 2: output_text.find('?b')]

                    subject = remove_extra_whitespaces(subject)
                    object = remove_extra_whitespaces(object)
                    relation = remove_extra_whitespaces(relation)
                    dataset.append(
                        {
                            'id': subject_counter,
                            'text': sentence_dict['text'],
                            'subject': subject,
                            'object': object,
                            'relation': relation
                        }
                    )
                    subject_counter += 1
                    # if subject_counter >= 6:
                    #     break

                    print('Subject counter: {}'.format(subject_counter))

            counter += 1
            print(str(counter) + " from 31363 (" , end='')
            print("%.2f" % ((counter / 31363) * 100), end='')
            print("%)")

            # if counter >= 29000:
            #     break
        except:
            error_counter += 1
            print('Error #{} occurred!'.format(error_counter))

results.write(json.dumps(dataset, ensure_ascii=False))
results.close()

```


در انتها، با استفاده از تابعی که فودمان تعریف کرده‌ایم (*remove_extra_whitespaces*)، اقدام به حذف فواصل اضافی در ابتدا و انتهای رشته‌ها می‌کنیم و تمامی روابط به‌دست آمده را در قالب یک انتیتی داخل لیست *dataset* ذخیره می‌کنیم.

```
def remove_extra_whitespaces(word):  
    if len(word) >= 1:  
        if word[0] == ' ':  
            word = word[1:]  
        if word[-1] == ' ':  
            word = word[:-1]  
  
    return word
```

در انتها اقدام به ذخیره‌ی تمامی انتیتی‌های دیتاست در درون یک جیسون واحد در فایل *sentences_with_relations.json* می‌کنیم.

فرمت هریک از موجودیت‌های فروبی به صورت زیر خواهد بود:

```
{  
    "id": 327,  
    "text": "شنا باعث کاهش آثار زیان‌بار استرس می‌شود",  
    "subject": "شنا",  
    "object": "کاهش آثار زیان‌بار استرس",  
    "relation": "باعث"  
},
```

```
{  
    "id": 82,  
    "text": "میلان در اولین دههٔ قرن، یکی از قدرتمندترین باشگاه‌های اروپا و جهان بود",  
    "subject": "میلان",  
    "object": "اولین دههٔ قرن",  
    "relation": "در"  
},
```

```
{  
    "id": 67,  
    "text": "مجموعه شعرهای احمد شاملو (دوجلدی) در آلمان غربی منتشر می‌شود",  
    "subject": "مجموعه شعرهای احمد شاملو (دوجلدی)",  
    "object": "آلمان غرب",  
    "relation": "در"  
},
```

همانطور که مشاهده می‌شود، توضیحات پیشین داده‌شده در مورد نتایج به‌دست آمده صدق می‌کنند و می‌بینیم که اکثر جملات به درستی آرگومان‌هایشان و روابط بین آن آرگومان‌ها استخراج شده است.

توجه: لازم به ذکر است که برای اجرا کردن این وابستگی جهانی روی جملات، با توجه به تعداد زیاد جملات و طولانی‌شدن فرایند، این کار در ۵ مرحله مجزا انجام گرفته و در نهایت، فروجی‌ها با همدیگر ادغام گشته‌اند و در قالب یک فایل میسون درآمده‌اند. فایل نهایی شامل مجموعاً ۲۳۹۲ روابط استخراج شده از جملات می‌باشد. طبیعتاً تعداد زیادی از جملات به دلایلی از جمله عدم وجود آرگومان، تک آرگومانی، نبودن موضوع در آرگومان اول و... حذف شده‌اند و مقادیری باقی مانده‌اند که بتوانیم در بخش‌های بعدی از آنها استفاده کنیم.

تصاویر زیر، نمایانگر مراحلی از اجرای کد مربوطه روی جملات هستند. تصویر اول پیشروی ریکوئست‌ها به ازای هر جمله را نشان می‌دهد و تصویر دوم نیز اطلاعاتی در مورد سرعت آپلود و دانلود و... در ازای هر بار دستیابی به مدل *Conllu* با استفاده از دستور اجرایی در *bash* می‌باشد. (این عملیات با صدا زدن *API* مربوط به سرویس *udpipe* انجام می‌گیرد.)

```
Run: connect_objects_and_relations x
↑ 21040 from 31363 (67.09%)
↓ 21041 from 31363 (67.09%)
↕ 21042 from 31363 (67.09%)
⏏ 21043 from 31363 (67.09%)
🗑 21044 from 31363 (67.10%)
21045 from 31363 (67.10%)
21046 from 31363 (67.10%)
21047 from 31363 (67.11%)
21048 from 31363 (67.11%)
21049 from 31363 (67.11%)
21050 from 31363 (67.12%)
```

```

Terminal: Local x +
[10/Jul/2022 04:12:22] "POST /api/seraji HTTP/1.1" 200 3244
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload  Total      Spent    Left     Speed
100  4794    0  3803  100   991    6703    1746  --:--:-- --:--:-- --:--:--   8484
[10/Jul/2022 04:12:22] "POST /api/seraji HTTP/1.1" 200 4325
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload  Total      Spent    Left     Speed
100  2805    0  2029  100   776    4362    1668  --:--:-- --:--:-- --:--:--   6058
[10/Jul/2022 04:12:23] "POST /api/seraji HTTP/1.1" 200 1913
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload  Total      Spent    Left     Speed
0         0     0     0     0     0     0     0  --:--:-- --:--:-- --:--:--    0

```

BERT ✓

ParseBERT چیست؟ ✓/۱

موج مدل‌های زبانی از پیش آموزش‌دیده، عصر جدیدی را در زمینه پردازش زبان طبیعی (*NLP*) آغاز کرده است و به ما امکان ساخت مدل‌های زبان قدرتمند را می‌دهد. در میان این مدل‌ها، مدل‌های مبتنی بر ترانسفورماتور مانند *BERT* به دلیل عملکرد پیشرفته‌ای که دارند، محبوبیت فزاینده‌ای پیدا کرده‌اند. با این حال، این مدل‌ها معمولاً بر زبان انگلیسی متمرکز هستند و زبان‌های دیگر را به مدل‌های چندزبانه با منابع محدود واگذار می‌کنند.

این مقاله یک *BERT* تک‌زبانه برای زبان فارسی (*ParsBERT*) پیشنهاد می‌کند که عملکرد پیشرفته آن را در مقایسه با سایر معماری‌ها و مدل‌های چندزبانه نشان می‌دهد. همچنین از آنجایی که حجم داده‌های موجود برای وظایف *NLP* در زبان فارسی بسیار محدود است، مجموعه داده‌ای عظیم برای وظایف مختلف *NLP* و همچنین پیش‌آموزش مدل تشکیل شده است.

ParsBERT در تمام مجموعه‌های داده، از جمله مجموعه‌های موجود و همچنین مجموعه‌های ترکیبی، امتیازهای بالاتری به دست می‌آورد و با پیشی گرفتن از *BERT* چندزبانه و سایر کارهای قبلی در وظایف تحلیل احساسات، طبقه‌بندی متن و شناسایی موجودیت نام‌گذاری، عملکرد پیشرفته را بهبود می‌بخشد.

۷/۲ استفاده از *ParseBERT* در زبان فارسی

دستاوردهای زیادی در زمینه زبان طبیعی برای زبان انگلیسی یا متی زبان های دیگر وجود دارد، اما در مورد فارسی، همانطور که می بینید، این دستاوردها زیاد نیست. شاید کمبود منابع داده ای، عدم افشای منابع توسط گروه های پژوهشی و جوامع غیرمتمرکز پژوهشی دلایل اصلی وضعیت فعلی فارسی باشد.

البته لازم به ذکر است که برخی از گروه های پژوهشی به این موضوع توجه دارند و نتایج، افکار و منابع خود را با دیگران به اشتراک می گذارند، اما همچنان به موارد بیشتری نیاز داریم.

زبان طبیعی ابزاری است که انسان ها برای برقراری ارتباط با یکدیگر از آن استفاده می کنند. بنابراین، مهم وسیعی از داده ها به عنوان متن با استفاده از این ابزار کدگذاری می شوند. استخراج اطلاعات معنی دار از این نوع داده ها و دستکاری آنها با استفاده از رایانه در حوزه پردازش زبان طبیعی (*NLP*) قرار دارد.

وظایف *NLP* مختلفی مانند شناسایی نهاد نامگذاری شده (*NER*)، تجزیه و تحلیل امسافات (*SA*) و پرسش/پاسخ وجود دارد که هر کدام بر جنبه خاصی از داده های متنی تمرکز می کنند تا به عملکرد موفقیت آمیز در هر یک از این وظایف دست پیدا کنند. در سال های اخیر روش های آموزش تعبیه واژه و مدل سازی زبان پیشنهاد شده است.

۷/۳ روش استفاده

در این بخش، روش مدل پیشنهادی ما ارائه شده است. این شامل پنج وظیفه اصلی است که سه وظیفه اول مربوط به مجموعه داده و دو وظیفه بعدی مربوط به توسعه مدل است. این وظایف عبارتند از جمع آوری داده ها، پیش پردازش داده ها، تقسیم بندی دقیق جملات، تنظیم قبل از آموزش و تنظیم دقیق.

۷/۳/۱ جمع آوری داده:

اگرچه چند مجموعه متن فارسی توسط دانشگاه لایپزیگ و دانشگاه سوربن ارائه شده است، اما جملات موجود در آن مجموعه‌ها از نظم منطقی در سطح مجموعه پیروی نمی‌کنند و تا مدودی اشتباه هستند. همچنین، این منابع تنها تعداد محدودی از سبک‌ها و موضوعات نوشتاری را پوشش می‌دهند.

بنابراین، برای افزایش کلیت و کارایی مدل از پیش آموزش‌دیده ما در سطوح کلمات، عبارات و جملات، لازم بود شکل جدیدی از پیکره را از ابتدا برای مقابله با محدودیت‌هایی که قبلاً ذکر شد، ایجاد کنیم. این کار با فزیدن در بسیاری از منابع مانند ویکی‌پدیای فارسی انجام شد.

۷/۳/۲ پیش پردازش داده:

پس از جمع‌آوری مجموعه پیش‌آموزشی، سلسله مراتب عظیمی از مراحل پردازش، از جمله تمیز کردن، جایگزینی، پاک‌سازی، و عادی‌سازی، برای تبدیل مجموعه داده‌ها به یک قالب مناسب میاتی است.

۷/۳/۳ بخش بندی متن:

پس از اینکه مجموعه از پیش پردازش شد، باید به جملات واقعی مربوط به هر سند تقسیم شود تا نتایج قابل توجهی برای مدل پیش‌آموزشی حاصل شود. یک جمله واقعی در فارسی بر اساس این نمادها تشریف‌ص داده می‌شود.

با این حال، تقسیم ممتوا صرفاً بر اساس این نمادها نشان داده است که مشکلاتی ایجاد می‌کند. در شکل ۲، نمونه‌ای از این مسائل نشان داده شده است. می‌توان مشاهده کرد که نتیجه شامل جملات کوتاه بی‌معنی و بدون هیچ گونه اطلاعات میاتی است، زیرا در زبان فارسی اختصاراتی وجود دارد که با علامت نقطه (.) از هم جدا شده‌اند.

به عنوان یک جایگزین، *Part Of Speech (POS)* می‌تواند راه‌حل مناسبی برای رسیدگی به این نوع خطاها و تولید فروجی‌های دلفخواه باشد. این روش سیستم را قادر می‌سازد تا رابطه واقعی بین جملات هر سند را بیاموزد.

Pre-training ۷/۳/۴

مدل ما بر اساس معماری مدل *BERT* است که شامل یک ترانسفورماتور دو طرفه چند لایه است.

مدل سازی نهایی ۷/۳/۵

مدل زبان نهایی (مدل پیشنهادی ما) باید با وظایف مختلفی تنظیم شود: تجزیه و تحلیل امساعات، طبقه بندی متن، و شناسایی موجودیت نامگذاری شده.

تجزیه و تحلیل امساعات و طبقه بندی متن متعلق به یک کار گسترده تر به نام طبقه بندی توالی است. تجزیه و تحلیل امساعات به عنوان یک وظیفه فاص طبقه بندی متن در بازنمایی امساعات پشت متن شناخته می شود.

پیاده سازی ۷/۴

برای پیاده سازی ، از کتابخانه *sentence_transformers* استفاده می کنیم. همچنین برای *cluster* بندی کردن، از کتابخانه *sklearn* استفاده می کنیم.

سپس با خواندن جمله های به دست آمده در مراحل قبلی پروژه ، جملات را می خوانیم و با انجام عملیات های زیر، جملات جدیدی تولید می شوند که در فایل *bert_sentence.json* مشاهده می شوند.

قطعه کد زیر مربوط به مدل بندی کردن با استفاده از *sentence_transformer* است.

```
30 def load_st_model(model_name_or_path):
31     word_embedding_model = models.Transformer(model_name_or_path)
32     pooling_model = models.Pooling(
33         word_embedding_model.get_word_embedding_dimension(),
34         pooling_mode_mean_tokens=True,
35         pooling_mode_cls_token=False,
36         pooling_mode_max_tokens=False)
37
38     model = SentenceTransformer(modules=[word_embedding_model, pooling_model])
39     return model
40
```

در نهایت با اجرای قطعه کد زیر ، عملیات *cluster* بندی کردن به اتمام می رسد و جملات جدید تولید می شوند.

```
64 # Perform kmean clustering
65 clustering_model = KMeans(n_clusters=num_clusters)
66 clustering_model.fit(corpus_embeddings)
67 cluster_assignment = clustering_model.labels_
68
69 clustered_sentences = [[] for i in range(num_clusters)]
70 for sentence_id, cluster_id in enumerate(cluster_assignment):
71     clustered_sentences[cluster_id].append(corpus[sentence_id])
72
73 for i, sentences in enumerate(clustered_sentences):
74     rtl_print(f'Cluster: {i + 1}', '20px')
75     rtl_print(sentences)
76     rtl_print('- - ' * 50)
77     bert_sentence.append({
78         'id': i,
79         'text': sentences
80     })
```

نمونه هایی از جملات تولید شده را در زیر مشاهده می کنید:

```
1 {
2     "id": 1,
3     "text": "آنان هرشب بر روی صحنه می رفتند و می درخشیدند اما به جای تشویق و دسته های گل، سنگ و کلوخ و ناصزا دریافت می کردند."
4 }
5 {
6     "id": 2,
7     "text": "امر هویدا در سال ۱۹۹۰ (میلادی) به آلمان پناهنده شد و مدت ۱۷ سال را در آنجا گذراند و در آن دیار به کارهای هنری خود ادامه داد."
8 }
9 {
10    "id": 3,
11    "text": "یک کمدی-درام به نویسندگی و کارگردانی الن سایدلر و مگان سایلر محصول سال ۲۰۰۹ است (And Then Came Lola: انگلیسی) و سپس لولا آمد."
12 },
13 {
14    "id": 4,
15    "text": "برای نمونه سیکلواکتاترانیید دی آنیون چنین است."
16 },
17 {
18    "id": 5,
19    "text": "را ملاقات می کنند و یک روز خاص را صرف با هم بودن می کنند، است؛ که توسط another parting او سپس با بازی در مینی سریال پنج قسمتی "
20 },
21 },
22 {
23    "id": 6,
24    "text": "و...وی در طول جنگ در عملیات های طریق القدس، فتح المبین، بیت المقدس، رمضان، والفجر ۴، بدر و والفجر ۸، حضوری فعال داشت."
25 },
26 },
27 {
28    "id": 7,
29    "text": "س خانواده پیامبر، به عنوان حافظ ایمان اسلامی و به عنوان تنها حاکمانی که تداوم بخش آرمان سلطنتی ایرانی بودند، حکمرانی می کردند."
30 }
```

Training ^۸

در نهایت برای *train* کردن داده ها، ابتدا با استفاده از قطعه کد زیر، به آن می گوییم که ۸۰ درصد داده هایمان برای *train* است و ۲۰ درصد برای تست:

```
1 from transformers import AutoConfig, AutoTokenizer, AutoModel
2 from sklearn import datasets
3 from sklearn.model_selection import train_test_split
4
5 wine = datasets.load_wine('../data/cleaned/cleaned_dataset.json')
6 X = wine.data
7 print(X.shape)
8
9 y = wine.target
10 print(y.shape)
11
12 dX_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
13 #80 percent training
14 #20 percent test
```

سپس ابتدا با *train* کردن ۸۰ درصد (*trainig*) روابط بین *entity* ها به دست می آیند.

برای قسمت تست، بین ۲۰ درصد جمله های خود که در واقع تقریباً ۶۰۰ تا می باشد، آن ها را با کد قسمت قبل که روابط را به دست می آورد و همچنین یک بار دیگر بر اساس *relation* های به دست آمده در *training*، و احتمال آن ها مناسبه می کنیم و در نهایت دقت و *loss* زیر به دست می آید:

trn_loss	val_loss	accuracy
1.22288	0.531899	0.895833
0.728065	0.22854	0.947917
0.521159	0.24	0.942708
0.436836	0.278884	0.9375
0.367163	0.296403	0.9375
0.299235	0.316672	0.9375
0.265763	0.326505	0.932292