Математички факултет,
Универзитет у Београду

Факултетски пројекат

Анализа сентимента из текстуалних података

Сташа Тонић
Алекса Кукркић
Жељана Товаришић

јануар 2023.

Садржај

1 Увод

Анализа сентимента представља веома користан приступ за аутоматизацију класификације поларитета датог текста. На пример, можемо одредити да ли се потрошачу свидео одређени производ на основу броја звездица које је оставио као оцену датог производа, услуге, делатности и сл. Међутим, основа дате оцене налази се у сировом тексту коментара који садржи мишљење купца. Анализа текста и категоризација сентимента различитих коментара, објава на блоговима, мејлова и осталих текстова, представља неоспорив алат компанијама за процену задовољства потрошача њиховим производима, услугама или самом компанијом. Анализа јавног мишњења, уз раме са *employer branding*-ом, важан је алат за формирање маркетиншке стратегије компаније и пласирање нових поризвода на тржиште.

IMDb база података, илити *Internet Movie Database*, последњих година је веома популаран извор за различите пројекте који се тичу анализе и визуелизације података. Комбинација оцена разних филмова од стране гледалаца и метаподатака представља веома интересантну подлогу и плодно тло за различита истраживања, те ћемо ми нашу демонстрацију анализе сентимента из текстуалних података извршити управо на овој бази подтака.

2 Анализа сентимента

Обрада текста један је најчешћих задатака машинског учења и његова примена може се поделити на три гране:

- Превод текста: превођење реченице са једног језика на други;
- Анализа сентимента: одређивање, на основу датог текста, да ли је сентимент према некој теми или производу позитиван, негативан или неутралан;
- Спам филтрирање: детектовање непожељних мејлова или порука.

Свака од ових примена веома је захтевна и укључује обраду веома велике количине текста, но ми ћемо се надаље фокусирати само на анализу сентимента.

Анализа сентимента, другачије називана и рударењем мишљења, представља технику обраде природног говора (Natural Language Processing - NLP) која се користи при закључивању да ли је сентимент неког текста позитиван, негативан или неутралан. Врло често је кроишћена од стране компанија приликом прикупљања повратних инфомрација од клијената и анализирања задовољства купаца неким производом или услугом, унапређивања пословања, истраживања тржишта, формирања будуће маркетиншке стратегије, разумевања потреба потрошача, истраживања репутације бренда и слично.

2.1 Типови анализе сентимента

Анализа сентимента је техника чији је примарни фокус на поларитету датог текста - позитиван, негатуван или неутралан, међутим, може ићи и изван тога и користити се за детекцију различитих емоција (срећа, бес, туга и сл.) или за предвиђање намера (нпр. да ли је купац заинтересован или не). Управо ће нам начин на који желимо да интерпретирамо повратну информацију клијената или циљ нашег истраживања дати поделу на неке од најпопуларнијих типова анализе сентимента:

2.1.1 Степенована анализа сентимента

Ако је некој компанији битна прецизност приликом детекције осећања потрошача према њиховим производима, на пример, жели да направи листу од 10 омиљених производа потрошача и објави је као водич куповине пред празнике, или пак жели да нађе 10 поризвода којима су купци најнезадовољнији и унапреди их, у том случају, може се послужити **степенованом анализом сентимента**. У овој методи поларитет садржи различите нивое позитивног и негативног, на пример:

- веома негативно;
- негативно;
- неутрално;
- позитивно;
- веома позитивно.

Ова скала може се користити код оцењивања звездицама тако што ће 5 звездица представљати ниво веома позитивно, а 1 звездица ниво веома негативно. По потреби, степеника у скали може имати више или мање, у зависности од тога колико желимо да наша анализа буде детаљна.

2.1.2 Детекција емоција

Ако бисмо желели да идемо корак даље од поларитета текста, онда је **детекција емоција** техника која нам може у томе помоћи. Овај тип анализе сентимента пружа нам могућност да уочимо различите емоције, попут среће, беса, љутње, туге и фрустрације. Једна од значајних примена ове методе јесте у селекцији кандидата за одређени посао, где на основу *CV*-ја или мотивационог писма можемо проценити расположење кандидата и његове особине.

Многи системи за детекцију емоција користе **лексиконе** који представљају листу одређених речи и емоције које оне интерпретирају. Неке речи су често коришћене приликом изражавања различитих емоција, на пример срећа и речи *пријатно* и *одлично* (”Изузетно укусна храна, пријатан амбијент и одлична услуга”). Једна од негативних страна употребе лексикона је што сви емоције исказују на другачији начин, те се одређене речи могу јавити и у позитивном и у негативном контексту, попут речи *никад* и *поново* (”Никад нисам јео бољу храну, сигурно ћу доћи поново” и ”Никад нисам имао гори интернет, поново има проблема иако је рутер замењен ”).

2.1.3 Анализа сентимента на основу аспекта

Ову методу користимо када желимо да знамо на који конкретно аспект или особину се одређени текст односи. Може послужити компанијама да открију на који конкретно аспект производа се односе негативни коментари, те унапредити само њега, уместо сматрања да је цео поризвод лош или знати који аспектом су потрошачи најзадовољнији. На пример, у коментару ”Лаптоп је фатнастичан, екран и камера су одлични, међутим, батерија јако кратко траје” би требало уочити да је негативан по питању трајања батерија датог лаптопа, али да је купац задовољан осталим аспектима.

2.1.4 Вишејезична анализа сентимента

Вишејезична анализа сентимента може се показати као врло компликована метода, с обзиром на то да укључује доста ресурса и јако пуно претпроцесирања . Већина потребних ресурса се могу пронаћи на интернету (попут језичких лексикона), међутим, и даље је потребно направити велики број од нуле (на пример језичке корпусе или алгоритме за детекцију шума). Ова метода се може комбиновати са алатом који аутоматски детектује и разврстава језике, након чега би се примењивала анализа сентимента текста на изабраном језику.

Вишејезични приступ од посебног је значаја за међународне компаније, где се коментари и утисци о производима могу наћи на више језика. Такође, важно је напоменути да приликом примене овог метода не постоји ниједан ”универзални алгоритам”, што значи да онај који функционише на српском, не мора функционисати на немачком.

2.2 Значај анализе сентимента

У новонасталој ери друштвених мрежа и борбе за слободу говора, где је постало никад лакше да мишљење појединца о нечему допре до шире народне масе, надгледање и праћење задовољства потрошача никад није било од већег значаја. Анализа сентимента постаје врло моћан алат коришћен у скоро свим врстама података.

Аутоматизовање анализе повратних информација клијената, попут интернет преписки, мејлова или одговора на различите упитнике, веома је значјно јер би за ручно анализирање и сортирање неколико хиљада коментара било потребно много више времена и енергија, а самим тим и новца, што овај метод чини веома великом погодношћу.

Тако да, неки од генералних бенефита анализе сентимента из текстуалних података су:

- Сортирање података у размери - као што смо напоменули, ручно сортирање велике количине података јако је напорно, посебно података различитих типова - твитова, мејлова, објава на Reddit-у и слично, те овај метод пружа најбрже и најјефтиније решење компанијама за обраду свих пословних података. Поред цене, овде и брзина игра велику улогу - брже уочавање негативних коментара и повратних информација омогућава и бржи одговор на њих, те помаже формирању везе и поврења између компаније и клијента.

- Анализа података у реалном времену - анализа сентимента нам може помоћи да уочимо проблеме у реалном времену, на пример, узевши у обзир то да овај метод има јако велику примену на друштевним мрежама, као ПР одређеног предузећа или јавне личности, можемо уочити кризу док се дешава и реаговатии што пре, не бисмо ли спречили да лоша објава о нашем производу, услузи или самом бренду, постане вирална и веома нашкоди имицу самог бренда. Један пример оваквог случаја јесте ситуација када је 1989. године Пепси је исплатио популарној певачици Мадони 5 милиона долара за право да користи њену нову песму *Like a prayer* у својој реклами и да се она сама појави у њој. Међутим, када се спот за ову нумеру појавио само неколико дана касније, постао је један од најконтроверзнијих музичких спотова икада и изазвао веома бурну реакцију јавности. Спот је био забрањиван, а Америчка породична асоцијација, као и многа црквена удружења, позивала су на бојкот Пепсија, што је довело до тога да су само неколико дана након објављивања рекламе морали потпуно да је повуку, изазвавши огромне новчане губитке, уз пад продаје. Помоћу аутоматизованих алата као што је анализа сентимента, овакве кризе је могуће уочити у раној фази, те што бржим деловањем утицати на ситуацију и спречити озбиљне новчане губитке. Слични примери могу се видети и у ситуацијама када су због само једног негативног коментара познате лилности о неком производу, цене деонице те компаније опадале астрономском брзином.

- Доследан критеријум - процењује се да се, приликом анализе сентимента одређеног текста, људи слажу у око $60 - 65\%$ случајева. Одређивање сентимента врло је субјективно и може зависити од карактера појединца, пређашњих искустава, уверења и тренутног расположења. Коришћење централизованог система анализе помаже предузећима да примене исти критеријум на све типове података, резултујући већом тачношћу.

Иако је анализа сентимента из текстуалних података врло користан алат, може наићи на неке потешкоће. Најчешћи проблеми са којима се ова метода сусреће јесу: емотикони, сарказам, ирониja и негација.

3 Припрема података и претпроцесирање

Један од најважнијих корака било које анализе података је њихова припрема и потребно претпроцесирање. Припрема података подразумева пречишћавање и трансформисање сирових података и врло често укључује њихово реформатирање, вршење одређених корекција и комбиновање различитих база података. *CrowdFlower*, компанија која поседује платформу за "обогаћивање података" намењену *data scientist*-има, спровела је истраживање које је укључивало 80 научника о подацима и открила да они проводе:

- 60% времена у организовању и пречишћавању података;
- 19% времена у прикупљању одговарајућих база података;
- 9% времена у рударењу података у циљу уочавања шаблона;
- 3% времена у тренирању модела;
- 4% времена усавршавајући алгоритам;
- 5% времена радећи неке друге задатке.

Као што је раније напоменуто, **IMDb** база података једна за NLP (*Natural Language Processing*) истраживања. Циљ проблема повезаног са овом базом података јесте детектовање да ли је критика неког филма позитивна или негативна. Припрема података за овакво истраживање укључује следеће кораке:

1. Токенизација - претварање реченица у речи или токене;
2. Уклањање знакова интерпункције;
3. Уклањање речи без одређеног значења - уклањање речи попут везника и речца (и, а, али, па, те и сл.);
4. Свођење речи на корен - речи је потребно свести на корен уклањањем флексија, најчешће префикса и суфикса;
5. Лематизација - још један приступ уклањању флексија одређивањем дела говора и коришћењем детаљне базе података датог језика.

3.1 Токенизација

Токенизација представља процес поделе реченице, параграфа или читавог текстуалног документа на мање компоненте, као што су речи или слова. Свака добијена мања компонента назива се **токеном**. Токени могу бити било шта - речи, слова, бројеви или чак и знаковни карактери који се формирају лоцирањем граница, које најчешће представљају белине.

Након што је токенизација извршена, у највећем броју слулајева следећи корак је формирање **вокабулара** или **речника** који мапира сваки јединствени токен у јединствени бројевни ID. Речник се потом користи за представљање датог текста као низа бројевних ID-јева, што даље могу користити различити модели машинског учења. Неке NLP библиотеке поседују опције за аутоматско креирање речника на основу датог текста.

Овај метод може бити коришћен за пребројавање речи неког текста или за одређивање учесталости одређене речи, тачније може нам омогућити да избројимо колико пута се одређена реч појављује у датом тексту.

Представимо ову методу на једном примеру. Посматрајмо реченицу

Ми радимо факултетски пројекат.

Токенизација на основу речи ће ову реченицу поделити на речи, најчешће на основу размака између њих.

[”**Ми**”, ”**радимо**”, ”**факултетски**”, ”**пројекат.**”]

Токенизација на основу подречи ће реченицу поделити на подречи.

[”**Ми**”, ”**рад**”, ”**имо**”, ”**факултет**”, ”**ски**”, ”**пројекат.**”]

Токенизација на основу карактера ће поделити реченицу на основу карактера.

[”**М**”, ”**и**”, ”**р**”, ”**а**”, ”**д**”, ”**и**”, ”**м**”, ”**о**”
”**ф**”, ”**а**”, ”**к**”, ”**у**”, ”**л**”, ”**т**”, ”**е**”, ”**т**”, ”**с**”, ”**к**”, ”**и**”
”**п**”, ”**р**”, ”**о**”, ”**ј**”, ”**е**”, ”**к**”, ”**а**”, ”**т**”, ”**.**”]

Токенизација представља темељ обраде природног говора зато што NLP модели обрађују текст на нивоу токена, тако да биле у питању основне NLP методе или напредни deep learning алгоритми, токенизација чини неизоставан корак као метод који текст читљив човеку претвара у компоненте читљиве машини.

Један од пакета погодан за вршење токенизације у R-у јесте пакет *tokenizers*.

3.1.1 Токенизација на основу речи

Токенизација на основу речи представља најзаступљенији вид токенизација и као што је већ поменуто, представља поделу датог текста на речи уочавањем задате границе. Граница може бити неколико, на пример белине и знакови интерпункције, што ће резултовати токенима различитих нивоа.

Када бисмо посматрали токенизацију на основу речи реченице

Why don’t you like statistics?

добили бисмо

[”**Why**”, ”**don’t**”, ”**you**”, ”**like**”, ”**statistics?**”]

Уочавамо да токени ”don’t” и ”statistics?” садрже знакове интерпункције. Ово може бити проблем зато што бисмо из реченице ”I love statistics.” добили токен ”statistics.”, што знача да ће наш модел научити две различите репрезентације речи statistics (statistics? и statistics.). Ако бисмо се одлучили за овакв принцип учења за наш модел, добили бисмо огромну количину речи коју модел треба да научи - свака реч у тексту у комбинацији са свим знаковима интерпункције. Овакав приступ је врло неефикасан, те узмимо интерпункцију у обзир.

Репрезентација

[”**Why**”, ”**don**”, ”**’**”, ”**t**”, , ”**you**”, ”**like**”, ”**statistics**”, ”**?**”]

може изгледати као очигледно решење, међутим боља репрезантеција речи ”don’t” би била ”do” и ”n’t”. Први разлог је што добијемо мањи број токена, а други што би модел у будућности, када би видео реч ”doesn’t”, дао токене ”does” и ”n’t”, јер се већ сусрео са ”n’t”, те би применио знање које је стекао.

Токенизација на основу речи у R-у може се имплементирати функцијом tokenize_words() из већ поменутог пакета ”tokenizers”. На нашој бази података то би изгледало овако:

```
tokenize_words(imdb_data$review)
```

Након извршене токенизације, модел свакој речи додељује ID који садржи повећи број информација о датој речи, пошто нам реченице обично пружају доста контекстуалних и семантичких информација.

Једна од главних мана овог метода јесте што као резултат даје огроман корпус који потом даје огроман вокабулар. Овај проблем се може решити ограничавањем броја речи које вокабулар може садржати. На пример, можемо сачувати само 5000 речи које се најчешће појављују у датом тексту (одређују се на основу учесталости речи у корпусу). Наш модел ће потом формирати ID за сваку од ових речи, а остале речи у тексту ће означити као речи ван вокабулара илити OOV речи (*Ou Of Vocabulary*). Међутим, ово ће довести до губитка информација пошто модел неће научити ништа о OOV речима. То може бити велики компромис за наш модел пошто ће научити једну исту репрезентацију за све OOV речи. Проблем OOV речи се може јавити и са другим речима које се уопште не појављују у тексту, на пример, може се десити да смо ми модел тренирали на скупу научних текстова, те он потом неће препознати речи из модерног говора или жаргонизме, пошто се они не појављују у научном говору.

Такође, проблем могу бити и речи које су погрешно написане. Ако се у тексту, на пример, појављују речи ”математика” и ”матемаитка”, наш модел ће речи која се касније појављује доделити OOV токен.

Проблем могу правити и хомоними - речи истог облика,а различитог значења, као у реченици ”Коса линија на папиру значи да јој коса није плава.", где се реч *коса* односи на врсту линије и именицу која означава длаке на нашој глави. Поред њих, незгоду чине и називи појмова који се састоје од две речи, на пример ”морски пас”, где би ова одвојена именица требало да буде третирана као један токен, а као резултат би дало два.

У енглеском језику се јавља и проблем контракција. На пример, ”should’t” је контракција од ”should not”, што модел врло вероватно неће препознати.

3.1.2 Токенизација на основу карактера

Као једна од алтернатива, јавља се **токенизација на основу карактера**. Токенизација на основу карактера дати текст дели на јединствене карактере. Идеја иза ове методе је да један језик има мноштво различитих речи, али ограничен број карактера, што као резултат даје јако мали вокабулар. На пример, у енглеском језику јавља се приближно 170 000 речи, а употребљава се само 256 карактера (слова, бројеви и специјални карактери). Овај метод може бити користан при проблемима попут идентификовања именованих

ентитета, где је фокус на проналажењу и класификовању одређених ентитета, уместо на специфичним речима коришћеним за њихово описивање.

У пакету "tokenizers" постоји функција tokenize_characters() која нам омогућава управо извршавање ове методе. Примена те функције, ако бисмо желели да токени буду појединачна слова, изгледала би овако:

```
tokenize_characters(imdb_data$review)[[1]]
```

Уместо јединице можемо уписати било који број који означава жељену дужину токена.

Једна од главних предности ове методе јесте што ће бити мање OOV речи јер ће модел моћи да креира репрезентацију речи које није видео током тренирања користећи репрезентације засебних карактера. Такође, предности токенизације на основу карактера јесу то што се може применити на бико који језик који користи исте карактере и што погрешно написане речи можемо написати исправно и тиме спречиti губитак информација који бисмо имали када бисмо их само маркирали као OOV речи.

Овакав врста токенизације јако је једноставно и може уштедетити много меморије и временске сложености, међутим, ни она није идеална. Разлог је што један карактер обично не садржи онолико информација, колико садржи реч, те може бити тешко формирати семантичке везе измешу речи и уочити смисао текста као целине. Поред губитка смисла, велики проблем је што овом методом знатно већи број токена него што бисмо токенизацијом на основу рече, на пример, за реч математичар бисмо уместо јегно токена добили 11. Ово може допринети знатном компликовању анализе.

Ипак, треба напоменути да постоје језици где један карактер носи доста информација, попут мандаринског, где се овај метод токенизације може показати врло практичним.

3.1.3 Токенизација на основу подречи

Један популаран метод токенизације јесте токенизација на основу подречи као спона између токенизације на основу речи и токенизација на основу карактера. Овај вид токенизације настао је у жељи да се реше стандардни проблеми који су сусрећу код ова два метода (веома велики речници, велики број OOV токена, различита значења сличних речи, токени који носе мало инофмрација...).

Ова метода ослаља се на два главна правила:

1. Речи које се често помињу у тексту не треба раздвајати на мање подречи.

2. Речи које се ретко појављују треба раздвојити на мање подречи.

На пример, реч "факултет" не треба раздвајати, док реч "факултети" треба раздвојити на "факултет и "и". Овим путем ће наш модел научити да се реч факултети формира користећи реч факултет, те ове две речи имају исти корен, али мало другачије значење. Слично, реч "токенизација" треба се раздвојити на "токен" и "изација", те ће модел бити истрениран да уочи да речи са истим кореном, као што су "токени" и "токенизација" имају слично значење. Такође, модел ће научити да речи које имају исте суфиксе, као што су "токенизација" и "модернизација" немају слично значење, али се користе у истим семантичким ситуацијама.

У токенизацији на основу подречи обично користи посебан симбол у циљу разликовања речи која чини почетак токена и оне која чини његов крај. Ратличити NLP модели користе различите симболе и они се могу додавати на почетну и на завршну речу токену. На пример, поменута реч ”токенизација” може се раздвојити на ”токен” и ”изација”, што би нам сигнализирало да је реч ”токен” почетни део нашег токена.

Овај метод за резултат даје вокабулар пристојне величине и јако је користан у сусрету са језицима компликованих морфолошких структура. Највећи број најсавременијих модела који су постигли изузетне резултате на енглеском језику користи управо овај вид токенизације. Неки од оваквих алгоритама су WordPiece, Unigram, Bye-Pair encoding и RoBERTa.

Као и за претходне методе векторизације и за векторизацију на основу подречи постоји функција у пакету ”tokenizers”. Она гласи овако:

```
tokenize_word_stems(imdb_data$review)
```

3.2 Уклањање знакова интерпункције

Један од веома важних корака у NLP-у јесте **уклањање знакова интерпункције** зато што може знатно упростити текст и олакшати алгоритмима машинског учења да уоче релевантне шаблоне у подацима. Знакови интерпункције као што су тачке, зарези и узвичниви уобичајено немају никакво семантичко значење, тако да могу бити уклоњени из текста без губитка битних информација. Такође, ово ће утивати и на смањивање нашег вокабулара, јер би иначе и знак ”!” добија свој ID исте тежине као и реч ”статистик”.

Постоји неколико начина за уклањање знакова интерпункције у R-у, а један од њих је коришћење функције removePunctuation() из памета ”tm”. Њена примена изгледала би овако:

```
imdb_data <- removePunctuation(imdb_data$review)
```

3.3 Уклањање речи без одређеног значења

Као што смо већ напоменули, претпроцесирање представља припремање текста тако да га машине могу употребити за разне задатке, попут анализа, предикција и слично. Током овог процеса, **уклањање речи без одређеног значења** представља један од важних корака. Ове речи обчно се називају стоп речима (stop words) и представљају речи које се најчешће појављују у неком језику. На пример, неке стоп речи у српском језику биле би ”зато”, ”па”, ”нити”, ”јер” и слично, а у енглеком ”the”, ”and” ”so” и ”what”.

У R-у постоји више начина да се стоп речи уклоне. Погледајмо како то можемо урадити у комбинацији са малопре већ поменутом функцијом tokenize_words(). Потребно је претходно инсталирти пакет ”stopwords”, потом, примена би изгледала овако:

```
tokenize_words(imdb_data$review, stopwords = stopwords::stopwords
  ("en"))
```

Зашто уклањамо ове речи? Њиховим уклањањем уклањамо речи које носе врло малу количину информација и фокусирамо се на оне значајне. Стоп речи могу створити шум, те отежати алгоритмима машинског учења проналажење релевантних шаблона. Поред тога што не немају скоро никакво семантичко значење, њиховим избацивањем знатно смањујем обим добијеног речника.

Међутим, овакве речи нису баш увек безначајне. Њихово уклањање у великој мери зависи од природе проблема који желимо да решимо. На пример, видимо да реченица

"I am not happy with this product at all.

након уклањања стоп речи постаје **happy product**

Тако да негативан коментар везан за неки производ заправо постаје позитиван. Када је у питању класификација, врло често елиминисање стоп речи неће бити потребно, те треба бити опрезан и прво проучити природу задатка који се пред нама налази, па након одреженог разматрања донети одлуку.

3.4 Свођење речи на корен и лематизација

Свођење речи на корен (*Stemming*) и лематизација представљају две технике NLP-а коришћене за свођење речи на корен или неку основу. Потреба за овим техникама јавља се у сусрету са варијантним облицима речи, тачније речима које су изведене из истог корена, али имају разлилите суфиксе или флексије. На пример, приликом претраживања, ако бисмо претражили реч "књиге", врло је вероватно да бисмо желели да видимо резултате претраге који садрже и реч "књига".

Свођење речи на корен представља процес уклањања префикса и суфикса неке речи да бисмо добили њен корен. Корен речи не мора нужно бити засебна реч или имати смисао сам по себи, али користи се за представљање свих облика једне речи. Узмимо за пример речи "токени", "токенизација" и "токенизовати". Заједнички корен свих ових речи биће "токен". Постоје различити алгоритми за извршавање овог процеса, али неки од најзаступљенијих у енглеском језику су *Porter Stemmer* и *Snowball Stemmer*.

Лематизација је пак процес свођења речи на основу или форму из речника, која се назима лемом. За разлику од свођења речи на корен, лематизација узима у обзир контекст и део текста у ком се одређена реч налази пре него је сведе на основу. Ово значи да ће лема увек бити нека смислена реч. Лема речи "токенизација" и "токени" биће такође токен, али лема речи "гори" биће реч лош. Кључ овог процеса представља лингвистика. Да бисмо добили смислену основу сваке речи, неопходно је размотрити морфолошко значење сваке од њих. То значи да нам је за извршавање овекве анализе неопходан речник сваког језика.

Да бисмо извршили ове методе, можемо употребити функције lemmatize_words() i stem_words() које се налазе у пакету "textstem". Њихова примена изгледа овако:

```
stem_words(imdb_data$review)
lemmatize_words(imdb_data$review)
```

Размотримо сада предности и мане сваког од ова два алгоритма.
Свођење речи на корен користи једноставна правила да уклони префиксе и суфиксе ради доласка до корена речи. Овакви алгоритми често су бржи од алгоритама лематизације ,

али су и мање тачни од њих пошто могу произвести корене који не представљају никакве засебне речи или су лоши у баратању са ирегуларним формама. Свођење речи на корен често се користи у претраживачима и у случајевима класификације текста када брзина игра битну улогу. Лематизација има већу тачност, али је и рачунски скупља метода. Често се примењује у напреднијим NLP задацима, попут генерисања природног говора, машинског превођења и анализе сентимента. Један од најпопуларнијих производа вештачке интелигенције данашњице, ChatGPT, користи обе ове методе.

4 Методи векторизације

За почетак, дефинишимо појам уграђивања речи. То је заправо техника претварања позитивних целих бројева у векторе фиксиране величине. Уопштено говорећи, то је техника у области NLP-а. Описује технику где су речи кодиране као густи вектори у високодимензионом простору и које носе неко значење. Свака реч има специфичну позицију унутар векторског простора. Штавише, тренирање модела доводи до тога да се семантички сличне речи појављују ближе једна другој у векторском простору.

4.1 Bag Of Words

Да бисмо анализирали текст и покренули алгоритме на њему, потребно је да текст представимо као вектор. Појам уграђивања једноставно значи да ћемо конвертовати улазни текст у скуп нумеричких вектора који се могу користити у алгоритмима. Постоји неколико приступа, а првенствено ћемо представити Bag Of Words.
Када користимо ову технику, примењујемо једноставну технику уградње речи. Технички гледано, узимамо цео наш корпус који је претходно обрађен и правимо огромну матрицу:
Колоне одговарају целокупном речнику који је икада коришћен са свим документима којима располажемо.
Линије одговарају сваком документу.
Вредност на свакој позицији одговара броју појављивања датог токена у оквиру датог документа.
Број којим попуњавамо матрицу је просто необрађени број токена у сваком документу. Ово се зове приступ термина френквенције или скраћено TF:

$$tf_{t,d} = f_{t,d}$$

где t означава појам, документ је означен са d, а f је сирови број.
Међутим, овај приступ више и није толико популаран. Па хајде онда да видимо која ограничења подразумева овај модел.
Што је реч учесталија, то јој придајемо већи значај унутар сваког документа. Међутим ово може бити проблематично јер уобичајене речи не доносе много информација о документу на које се односе. Другим речима, речи које се највише појављују нису најинтересантније за издвајање информација из документа. Дакле, можемо искористити чињеницу да речи које се појављују ретко доносе много информација о документу на које се односе.
Хајде сада да видимо како ова техника ради.

Први корак:

Узмимо за пример три реченице: "Он је добар дечак", "Она је добра девојчица", "Дечак и девојчица су добри".

Други корак: Претпроцесирање

У овом кораку ми вршимо - уписивање реченице малим словима и уклањање тзв. Stopwords (заустaвних речи). Дакле, вршимо токенизацију, затим лематизацију, а затим издвајамо речи из сваке реченице. Из прве "добар"и "дечак", из друге "добра"и "девојчица"и из треће "добри", "дечак"и "девојчица".

Трећи корак је примена Bag Of Words операције

То ћемо да урадимо на следећи начин:
У првој реченици: добар = 1, дечак = 1, девојчица = 0.
У другој реченици: добра = 1, девојчица = 1, дечак = 0
У трећој реченици: добар = 1, девојчица = 1, дечак = 1

У овом кораку смо свакој речи доделили вредност у складу са њеним понављањем у реченици.
Ако је реч присутна два пута у истој реченици, тада тој речи додељујемо вредност 2.

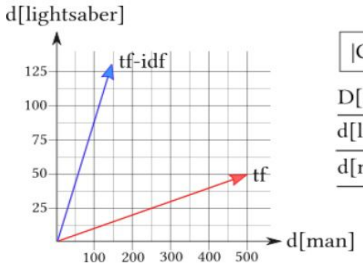
4.2 TF-IDF

TF-IDF је скраћеница од Term Frequency Inverse Document Frequency.
Ове методе су биле прилично популарне дуго времена, пре напреднијих техника као што су Word2Vec или Universal Sentence Encorder.
У TF-IDF уместо да попуњавамо BOW матрицу сировим бројевима, једноставно је попуњавамо термином фреквенција помноженим са инверзном фреквенцијом документа. Намера му је да одрази колико је реч важна за документ у збирци или корпусу:

$$tfidf_{t,d} = f_{t,d} \times invf_{t,d} = f_{t,d} \times (log\frac{M}{df_t})$$

где f(t,d) рачуна број појављивања t у d;
df(t) рачуна број докумената који садрже реч t;
M означава укупан број докумената у корпусу.
Такође, ова метрика дозвољава дефиницију фактора логаритамске тежине idf-a.

На следећој слици приказаћемо ефекат tf-idf-a:



|C|=2e3

D[StarWars]	tf	df	idf	tf-idf
d[lightsaber]	50	6	2.52	126
d[man]	500	1e3	0.30	150

Вредност TF-IDF расте пропорционално појављивању речи у TF, али ефекат је уравнотежен појављивањем речи у сваком другом документу (IDF).
Оба ова метода репрезентују један документ као један вектор.
Највећа предност овог метода је његова екстремна ефикасност у поређењу са релативно малим напором потребним за његову имплементацију. Понашање кодирања је строго синтаксичко.
Недостатак овог приступа је недостатак простора за уградњу као и потпуно занемаривање реда речи који претпоставља модел.

4.3 LSA/I

LSA/I је скраћеница од Latent Semantic Analysis/Indexing. Популарна метода за смањење векторског простора је примена Гаусове елиминације, која се још назива и метода "смањења редова". Најпре је потребно да се сви вектори документа комбинују у матрицу, где је сваки вектор документа једна колона.
LSA користи мало софистицираниј у, нејасну верзију Гаусове елиминације под називом "Декомпозиција сингуларне вредности". Ову методу је први увео Deerwester 1990. године и стекао је велику пажњу својим математички мотивисаним приступом за стварање "густо насељеног"латентног семантичког простора.

4.4 Мерење сличности између докумената

У векторском простору, скуп докумената одговара скупу вектора у векторском простору. Косинусна сличност описује сличност између два вектора према косинусу угла у векторском простору:

$$cos_{sim}(d_1, d_2) = \frac{\bar{V}(d_1)\bar{V}(d_2)}{\mid \bar{V}(d_1) \mid\mid \bar{V}(d_2) \mid}$$

Наведимо сада пример мерења сличности користећи tf-idf:
Користићемо веома моћан пакет, пакет text2vec.
Почињемо од претпроцесирања. Прво ћемо сва слова претворити у мала слова, а затим уклонити карактере који нису алфанумерички.

```
require(stringr)
require(text2vec)
prep_fun = function(x){
  x %>%
    #pretvaramo tekst u mala slova
    str_to_lower %>%
    #sklanjamo karaktere koji nisu alfa-numericki
    str_replace_all("[^[:alnum:]]", " ")
}
clean_text = prep_fun(text)
```

Затим можемо креирати речник и изградити матрицу термина докумената за корпус. Речник се састоји од јединствених појмова у корпусу.

```
it = itoken(clean_text, progressbar = FALSE)
v = create_vocabulary(it)
vectorizer = vocab_vectorizer(v)
dtm = create_dtm(it, vectorizer)

# Tf-idf tezine
tfidf = TfIdf$new(smooth_idf = TRUE)
dtm_tfidf = fit_transform(dtm, tfidf)
```

Косинусна сличност одређује се помоћу следеће функције:

```
tfidf_cos_sim = sim2(dtm_tfidf, method="cosine", norm="l2")
print(tfidf_cos_sim)
```

Резултат показује да је сличност између две наведене реченице 1, што указује на то да су потпуно исте, што ипак није случај, јер очигледно у другој реченици имамо још једну реч, реч too.
На овај проблем наилазимо због IDF функције. Када год се један термин појави у свим документима осим у једном, овај термин неће давати никакву тежину користећи подразумевану IDF функцију у text2vec.
Наш пример може бити посебан случај где имамо само две реченице које се разликују у само једној речи и такве проблеме управо решавамо одабиром одговарајуће IDF функције.

```
TfIdf$private_methods$get_idf = function(x) {
  cs = colSums( abs(sign(x) ) )
  if (private$smooth_idf)
    idf = log(nrow(x) / cs + 1)
  else
    idf = log(nrow(x) / (cs))
  Diagonal(x = idf)
}
```

Користећи ову нову IDF функцију, хајде да видимо која је сада косинусна сличност.

```
tfidf = TfIdf$new(smooth_idf = TRUE)
dtm_tfidf = fit_transform(dtm, tfidf)
tfidf_cos_sim = sim2(dtm_tfidf, method="cosine", norm="l2")
print(tfidf_cos_sim)
```

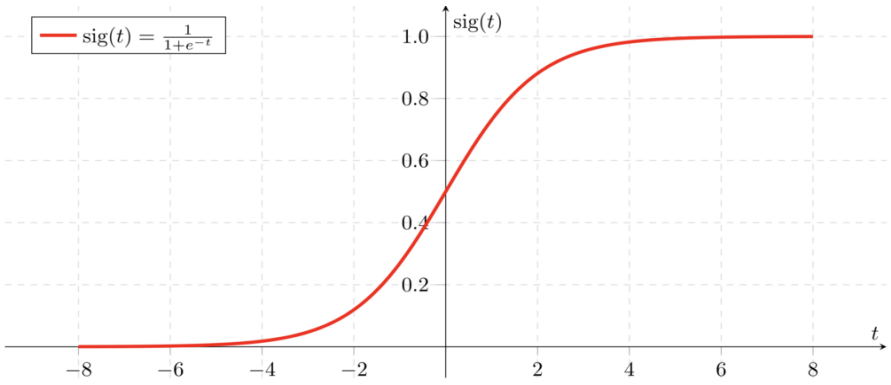
Видимо да сада реченице имају сличност 0.74 што има много више смисла.

5 Прављење модела

Након одрађеног потребног претпроцесирања и векторизације, прелази се на последњи корак у анализи сентимента, прављење модела. Подаци се деле на тренинг и тест скуп. Модел се креира на тренинг скупу, и затим се на основу њега могу оценити вероватноће на тест скупу. У конкретном примеру са IMDB базом циљ је предвидети да ли је критика филма позитивна или негативна.
Једни од најчешће коришћних алгоритама машинског учења за класификацију, укључујући и анализу сентимента, су Logistic Regression и Naive Bayes.

5.1 Logistic Regression

Логистичка регресија је класификација која се најчешће примењује за решавање проблема бинарне класификације, резултат је обично 0 или 1 (тачно или нетачно, да или не). Користи се када је зависна променљива категоричка. Она користи Сигмоидну функцију која која служи да било коју реалну вредност преслика у вероватноћу, тј. у другу вредност из интевала [0,1].



Логистички регресиони модел је облика:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

За прављење овог модела у R-у се користи функција glm().

Наравно, постоји и вишеструка логистичка регресија, то је случај када постоје више од две могуће класе зависне променљиве. Овакав модел у R-у се креира помоћу функције multinom().

5.2 Naive Bayes

Naive Bayes je još jedan klasifikacioni algoritam koji se koristi za klasifikaciju teksta sa obimnim trening skupom. Pogodan je za analizu sentimenta, kategorizaciju dokumenta, filtriranje neželjenih email-ova, itd. Sam naziv "Naivni Bajes"potiče od toga što se naivno pretpostavlja da je pojava određenih obeležja nezavisna od svake druge, što u realnosti nije slučaj. On je zasnovan na Bajesovoj teoremi koja daje uslovnu verovatnoću događaja A, ako se dogodio neki drugi događaj B:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

6 Закључак

Анализа сентимената пружа начин за разумевање ставова и мишљења изражених у текстовима. Главни циљ пројекта је био да се изгради модел који помаже у што бољем класификовању мишљења (нпр. рецензије филмова).
У овом раду истражили смо како приступити анализи сентимената у тзв. tidy структури података где су најпре детаљно одрађене технике претпроцесирања, односно чишћења и обраде текстуалних података. Један од битнијих задатака анализе снетимента је спознаја које су речи од кључног значаја за одређени текст. За то су коришћене методе векторизације. На крају, над добијеним подацима, односно на тренинг скупу, креиран је модел логистичке регресије и оцењена је његова тачност.

Литература

[1] Metzger, K., Sader Fosalaie, A. & Yonan, N. (2018). *Text mining - Sentiment analysis*. Berlin: Humboldt-Universitat zu Berlin.

[2] *Sentiment Analysis: A Definite Guide*. (2022). Preuzeto sa https://monkeylearn.com/sentiment-analysis/

[3] Nabi, J. (2018). *Machine Learning - Text Processing* Preuzeto sa https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958

[4] Abhimanyu, S., Chaitanya, K., Necati, A. *Sentiment Analysis* Preuzeto sa https://chaitanya1731.github.io/img/prj-1/report.pdf

[5] Bryan, T. (2020).*Performing Sentiment Analysis on Movie Reviews* Preuzeto sa https://towardsdatascience.com/imdb-reviews-or-8143fe57c825

[6] Khanna, C. (2021.). *Word, Subword and Character-Based Tokenization: Know the Difference* Preuzeto sa https://towardsdatascience.com/word-subword-and-character-based-tokenization-know-the-difference-ea0976b64e17

[7] Khanna, C. (2021.). *Text pre-processing: Stop words removal using different libraries* Preuzeto sa https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a

[8] *What is the difference between stemming and lemmatization?*. (2021). Preuzeto sa https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/