Szövegbányászat

Bevezető gondolatok a szövegbányászatról

Az emberek, már a kezdeti, ősi civilizációkban is, a szóbeli mellett, jellemzően írásbeli szövegek segítségével tárolták, és adták át egymásnak az ismereteiket. Napjainkban a rögzített tudásanyagainknak egyszerű szöveges dokumentumokban található. Ezt a feltevést támasztják alá, többek között a Merill Lynch elemzései is, melyek becslése szerint az üzleti információk körülbelül 85%-a található strukturálatlan, illetve gyengén strukturált szövegekben. Az általunk kezelt szövegek növekvő arányban digitálisan tárolt dokumentumok.

Így talán nem meglepő, hogy a szövegek, és főképp a bennük lévő információk, kezelésének hatékonyabbá tétele, napjaink egyre fontosabbá váló informatikai tevékenysége. Az egyik tudományág mely ezzel foglalkozik, a szövegbányászat.

A szövegbányászatot definiálhatjuk úgy, mint szöveges adatokon végzett feldolgozási és elemzési tevékenység, mely célja a dokumentumban rejtett információk feltárása, azonosítása, és elemzése. A szövegbányászat interdiszciplináris szakterület, mely olyan informatikai eszközök mellett, mint a gépi tanulás és a hatékony algoritmusok, a matematika és a nyelvészet eszközeit is felhasználja.

A szövegbányászat két nagy alaptípusa a keresés és a rendszerezés. A keresésnél kiválasztjuk azokat a dokumentumokat, ahol egy adott keresőkifejezés előfordul, míg a rendszerezésnél valamilyen kategóriákba, vagy előre nem definiált csoportokba soroljuk azokat. A szövegbányászat főbb feladattípusai is ilyen jellegű feladatokat, vagy ezek kombinációját hajtják végre céljaik elérése során.

Ezek a főbb feladattípusok a kereséstámogatás és információ-visszakeresés, az Információkinyerés, az osztályozás, a csoportosítás, az összegzéskészítés, a kivonatolás, a válaszkereső rendszerek, a szövegelemzés, és a napjainkban egyre inkább tért nyerő webes tartalomkeresés. Ezekről külön alfejezetekben részletesebben is írok, de csak a dolgozat szempontjából leginkább hangsúlyos információkinyerést fogom mélyebben részletezni.

Szövegbányászati előfeldolgozás

Számos szövegbányászati feladat megoldható, már létező, adatbányászati eszközökkel, és algoritmusokkal. Ehhez a szöveges adatokat úgy kell transzformálni, olyan alakra kell hozni, hogy ezek a bejáratott eljárások, lehetőleg hatékonyan, képesek legyenek működni rajta. Ezt a transzformációs lépést szokták előfeldolgozásnak is nevezni. A folyamat végeredménye a dokumentumot reprezentáló modell. Az előfeldolgozás egységesítési, formalizációs és normalizációs feladatokat is tartalmaz.

A dokumentum alapvető jellemzői

dokumentumot hordozó médium:

Ha digitális, gondoskodni kell az elérhetőségről, ha pedig analóg, akkor előbb digitalizálni kell. ha képi, akkor a képen látható szövegeket, ki kell nyerni, lehetőleg automatikus karakterfelismeréses vagy akár manuális begépeléses módszerrel.

A dokumentum elérési helye:

A hagyományos világban lehet egy könyvtár egy polca, míg a digitális világban jellemzően lehet egy URL (Uniform Resource Locator) mely egy egységes erőforrás-azonosító és az interneten keresztüli adatelérésre szolgál, vagy lehet egy háttértárolón egy fájl elérési útvonala. Szövegbányászatnál mindenképpen fontos, hogy rendelkezzünk minimum olvasási jogokkal az adott dokumentumon.

Mérete

A nyers digitális állományok legelterjedtebb általános méretmegadási módja az adattárolón elfoglalt terület alapján, bájt alapon történik. A szövegfájlok mérete megadható még benne szereplő karakterek, szavak vagy az oldalaik számával. A szöveges korpuszok méretét a benne szereplő dokumentumok összesített mérete adja.

Az előfeldolgozási fázis után a méretet jellemzően szavaik, ritkábban karaktereik számával adhatjuk meg. A modellbeli reprezentációjuk méretét pedig olyan a felírásukra használt elemek számával jellemzik, mint a tokenek vagy az egyedi szavak száma, esetleg a modell indextömbjének mérete.

statisztikai jellemzői

A későbbi feldolgozás során szükségünk lehet egyes statisztikai adatokra is a dokumentumunkról. A legfontosabb ilyen jellemző lehet a szavak eloszlása, de esetenként fontos lehet még a karaktereinek eloszlása, a szóhosszoknak az eloszlása vagy átlaga. A magán és mássalhangzók száma és eloszlása, vagy a mélyebb nyelvi elemzéssel kideríthető zöngés és zöngétlen magán- illetve mássalhangzók száma, melyekből például következtetni lehet az adott dokumentum nyelvére.

Metaadatai

A dokumentum adathordozón való tárolásakor keletkeznek metaadatok, mint a keletkezési ideje, az utolsó módosítás ideje, a fájlnév, illetve a hozzáférést módosító attribútumok.

Szövegbányászati szempontból egyéb jellemzők is fontosa, ilyen például a dokumentum tartalmára vagy a keletkezési körülményeire vonatkozó metaadatok, mint a nyelv, a keletkezési idő/hely, a dokumentum témája vagy más szempont szerinti besorolása, a stílusa, a nyelvezete vagy zsánere. Ezek osztályozáskor vagy csoportosításkor lehetnek segítségünkre. Ezek közül az adatok közül nem áll rendelkezésünkre minden, de néhányat, ha hiányzik, szövegbányászati eszközökkel meg tudunk határozni. A nyelv vagy a téma is lehet ilyen.

Formátuma

A dokumentum formátumát többnyire a készítésére használt eszköz típusa bekorlátozza. egyes eszközök egyszerű szövegként (plain text, txt) mentenek, mígy másik eszközök formátumai különböző formázási és szerkezetre vonatkozóadatok, de akár a dokumentum régebbi változatai is lehetnek. Az ilyen tárolási formátumok, a teljesség igénye nélkül, például a doc/docx, a pdf, rtf, de akár a HTML és az XML is ide sorolható.

A dokumentum készítői által kiemelt szövegrészek lehetnek külön formázásúak, mint például a félkövéren szedett, a dőlt, vagy az aláhúzott szövegrészek, és lehetnek szerkezetileg meghatározó helyen, mint a különböző szintű címekben, vagy például a webes elérésű dokumentumok az elérési útvonalában is szerepelhetnek. A dokumentumban kiemelt kifejezések a dokumentum szempontjából relevánsabbak lehetnek, ezért érdemes lehet, ezeket az iformációkat a modell reprezentációba is átvinni.

Karakterkódolása

Az általunk használt adatok különböző karakterkódolásokban fordulhatnak elő, hiszen a számítógép, a szöveges adatokat is kettes számrendszerű számokként tárolja, ezért szükséges annak a meghatározása, hogy egy-egy szám milyen karaktert kódol.

Az egyik első széles körben használt kódtábla az ASCII (*American Standard Code for Information Interchange*), amit az Amerikai Egyesült államokban alkották meg, először hét biten ábrázolt, ami 128 karakter ábrázolását teszi lehetővé, melyek némelyike vezérlő karakter, mint az új sor, vagy a tabulátor, illetve vannak nem látható vezérlő karakterek is melyek a korabeli nyomtatókat vezérelték. Az ASCII 8. bitjét eleinte paritásbitnek szánták, de később általánossá vált, hogy a karakterkészlet kibővtésére használták, az így nyert további 128 lehetséges állapottal az [ISO 8859-1](http://hu.wikipedia.org/wiki/ISO_8859-n) Latin-1-es kódolás a nyugat európai, míg az [ISO 8859-](http://hu.wikipedia.org/wiki/ISO_8859-n)2 inkább a kelet európai nyelvek speciális ékezetes karaktereit írja le. Napjainkban egyre inkább az Unicode veszi át a vezető karakterkódolási szabvány szerepét, mely arra hivatott, hogy legyen képes az ismert nyelvek összes karakterét kódolni. Legelterjedtebb megvalósítása az UTF-8, mely változó hosszon kódolja a karaktereket, ezért tömör kódolást biztosít, mindeközben kompatibilis a 7 bites ASCII-vel. A karakterkódolások különbözősége mellett olyan dolgok is nehezíti a feladatunkat, mint hogy a a sor végének jelölése egyes platformokon eltérhet.

Az ismert kódolások részben, vagy teljesen inkompatibilisek, ezért a konvertálásnál kiemelt kérdés a karakterkódolás, mivel ennek a helytelen használata, a beolvasott szöveg torzulásához vezethet, vagy az akár teljesen használhatatlanná is válhat.

A karakterkódolási információt esetenként maga a fájl tartalmazza, ha nem ismerjük, akkor az előfeldolgozás helyessége nem garantálható.

Modellalkotás

Az ismertebb dokumentum reprezentációs modellek bemutatása

A legelterjedtebb modellek jellemzően valamilyen numerikus objektumok. Ez számos előnnyel jár, mint a kisebb tárolási méret. Ugyanis ha a szavakat karakterenkénet letároljuk, és jellemzően egy-két bájtos egy karakter, akkor nagyobb helyet vennénk igénybe, mint a numerikus tárolásnál, ahol szavanként egy darab 2 vagy akár 4 bájtos számmal számolunk. Mindemellett, a számok használatának van egy másik jelentős előnye, méghozzá az, hogy matematikai műveleteket, és transzformációkat hajthatunk végre az ilyen modelleken. Ráadásul a modellben a matematikai eszközökkel való munka elősegíti a dokumentumok hatékony kezelését.

Azt, hogy ténylegesen milyen modellt és adatábrázolást használunk befolyásolja a megoldandó feladat típusa. Keresés jellegű feladatoknál egy megfelelő szóelőfordulás táblázat is nagy szolgálatot tehet, míg a rendezés jellegű feladatoknál összetettebb dokumentum összehasonlító módszerekre van szükség.

A modellalkotásnál használt három nagy matematikai elméleti megközelítés a halmazelméleti, az algebrai, illetve a valószínűségelmélet alapú. A halmazelmélet alapú modellek jó szolgálatot tehetnek az egyes keresőrendszerekben, hisz kereséskor minden dokumentumra fennáll, hogy része az eredményhalmaznak, vagy sem. Az algebrai modellben a dokumentumokat olyan algebrai objektumokként reprezentáljuk mint a vektor, vagy a mátrix. Ezeket algebrai műveletekkel össze is hasonlíthatjuk, ezért ezek már használhatók, rendszerezési feladatok megoldásánál is. A legelterjedtebb megvalósítása a vektortér modell és annak változatai. A valószínűségi modellben maguk a dokumentumok valószínűségi események által reprezentáltak, míg a kapcsolataik feltételes valószínűségi becslések eredménye. Ezek mellett léteznek komplex függvénytani eszközöket alkalmazó spektrális szövegbányászati modellek is.

A vektortér modell

A vektortér modellben hatékonyan meg lehet határozni a dokumentumok távolságát, illetve hasonlóságát. A szövegbányászatnál gyakran élünk különböző intuitív heurisztikákkal, melyek meghatározzák, az ezeket felhasználó eszközök felhasználhatósági körét és korlátait. A vektortér modellnél azt jelentjük ki intuitív módon, hogy azokat a dokumentumokat tekintjük hasonlónak, melyek szókészlete átfedi egymást, és ennek a hasonlóságnak a mértéke arányos az átfedés mértékével. A modell egy sokdimenziós vektortérben, vektorokkal reprezentálja a dokumentumokat. A vektortérben az egyes dimenzióit a dokumentumgyűjtemény egyedi szavai adják. Tehát egy-egy dokumentum a szavaiból álló vektor, abban a vektortérben, ahol az egyes szavak a téret kifeszítő vektorok. A dokumentumgyűjteményt a szó-dokumentum mátrixszal reprezentáljuk. Az egyedi szavak összessége a szótár, vagy más néven lexikon.

Általában a teljes szótár csak egy kis része fordul elő egy-egy dokumentumban, ezért a szó-dokumentum mátrix ritka. Mindemellett az egyedi szavak száma meglehetősen nagy, akár több százezres, vagy akár milliós nagyságrendet is elérheti. Ezért a mátrix méretét nyelvtechnológiai, illetve matematikai eszközökkel érdemes lehet csökkenteni. A vektortérmodell másik nehézsége az alaphipotézisből származik, miszerint a szavak megléte, illetve száma adja a dokumentumok hasonlóságát. De nem tér ki, és nem is tárolja a szavak sorrendjére se a szövegen belüli pozícióira vonatkozó információkat. Ezért hívják szózsákmodellnek is. Ez a modell a feladatok nagy részében jól használható, egyszerű és hatékon megoldást nyújt. A szósorrend elvesztéséből való hátrányát sokszor más modellekkel való együttes alkalmazásával küszöbölik ki.

A szó-dokumentum mátrix jellemző súlyozási sémái

A legegyszerűbb módszer, ami csak a szó dokumentumbeli esetleges meglétét jelöli, a bináris reprezentáció. Ahol a mátrix egy adott dokumentumot reprezentáló oszlopvektorában, egy adott szóhoz tartozó sorban nullát írunk, ha az adott szó nem szerepel a dokumentumban és egyet, ha igen, és nem számít az, hogy hányszor, ez az információ elvész. Egy másik egyszerű módszer, ha a súly maga az előfordulások száma. Itt viszont felvetődhet a kérdés, miszerint tényleg lináris-e a függés a szó adott dokumentumra vonatkozó relevanciája, és az adott szó előfordulásainak száma között. Tehát egy jóval gyakrabban előforduló szó tényleg ugyanilyen mértékben fontosabb is. Az esetek nagy többségében a válasz egyértelműen nem, sokkal nagyobb megkülönböztető értéke van, ha az adott szó előfordul a dokumentumban, mint annak, hogy az tízszer, vagy százszor fordul elő. Ezt kiküszöbölendő kezdték el használni a logaritmikus súlyozást. Ezen egyszerű súlyozási sémák nagy hátránya, hogy nem veszik figyelembe az egyes dokumentumok hosszát. Pedig feltételezhetően egy rövidebb dokumentumban egy szó bizonyos számú előfordulása fontosabb, mint az ugyanekkora számú előfordulása egy jóval hosszabb szövegben. Így a bonyolultabb súlyozási függvények figyelembe veszik a szó dokumentumbeli előfordulási gyakoriságát is. Ezt a gyakoriságot az angol irodalomben term frequency kifejezéssel illetik, és a szakirodalmakban gyakran TF-nek rövidítik. Ezen elven alapuló jellemző megvalósítások a gyakoriság alapú, illetve a normalizált logaritmikus súlyozások. Ez a megközelítés már közelebb áll a valósághoz, de nem veszi figyelembe azt, hogy vannak olyan szavak, melyek minden dokumentumban egységesen sűrűn fordulnak elő és nincsenek semmilyen összefüggésben az adott dokumentumok tartalmával. Az angol terminológiát átvéve ezeket stopszavaknak hívjuk. Ezért a szó fontosságát befolyásolhatja a teljes gyűjteményben való előfeldolgozásainak száma is, amit szoktunk gyűjteménytámogatottságnak is nevezni, vagy angolul collection frequency az elterjedt megnevezése. Fontos lehet továbbá az előfordulások eloszlása a korpuszon, ugyanis a koncentráltan kevés dokumentumban, de azokon belül nagy számban szereplő szavak valószínűleg fontosabbak is az adott dokumentumokra nézve, illetve nagyobb a dokumentumok közti megkülönböztető szerepük is. Ha nk-val jelőljük a tk szót tartalmazó dokumentumok számát, és N pedig a dokumentumok száma a korpuszban. Akkor a dokumentum gyakoriság a df = nk/N képlettel számolhatjuk. Ez a szám megadja, az adott szó ritkaságát, azt hogy a szó megléte, és annak száma mennyire különbözteti meg a dokumentumokat egymástól az adott korpuszban. Az idf (inverse document frequency) súlyozási sémában ennek inverzével számolunk. Maga a dokumentumgyakoriság inverzét számoló egyik elterjedt súlyozási függvény:

idf(tk) = log(N/nk)

A leggyakrabban használt tk-idf (term frequency and inverse document frequency) súlyozást úgy kapjuk, hogy a vektortérmodell szavakat reprezentáló tengelyeit az adott szavak idf által megadott relevanciájával arányosan súlyozzuk.

Tehát a szó-dokumentum mátrix adott dik súlya kiszámítható a

dik= fki \* idf(tk) képlettel

E mellett számos más képlete, és módosulata létezik, amikre most nem térek ki.

A tf-idf súlyozás értéke magas lesz a nagy megkülönböztető képességű, adott dokumentumra gyakori, de a korpuszra ritka szavaknál. Alacsonyabb lesz a korpuszban gyakoribb, vagy az adott dokumentumban ritkább szavakban, és elhanyagolhatóan alacsony, akár zérus, az olyan szavakban melyek az egész korpuszban gyakran fordulnak elő.

Módok a szöveg felbontására, illetve a szótár felépítésére

Mielőtt felírhatnánk a választott modellbe a dokumentumainkat, számos azt megelőző előfeldolgozási lépésre lehet szükségünk. Az első ilyen lépés a reprezentációs egységekre való bontás. Ez leggyakrabban egyszerű szavakra való bontás, de egyes esetekben, mint például a kivonatolás a dokumentumok bekezdésekre történő, illetve mondat szintű felbontására is szükség van. Ezután a vektortér modell használata előtt meg kell határozni a szótár szavait is. Az összes egyedi szó szövegekből való kinyerése után, célszerű lehet ezek egy kis fontosságú, de esetenként sok szót tartalmazó, részét elhagyni, ezzel jelentősen csökkentve, az indexelt szavak számát és a vektortér méretét.

A strukturális szegmentálás

Dokumentumaink szövege számos hierarchiaszintbe sorolható. Ilyen strukturális egységek lehetnek például a kötet, rész fejezet, szakasz, pont, illetve a bekezdés a mondatok és a szavak. Ezek és az ezekből összeállítható dokumentumtérkép automatikus meghatározása egyes fájlformátumokból kinyerhető információk alapján viszonylag egyszerű feladat is lehet, míg például az egyszerű szövegekből nehézkes, és esetenként koránt sem egyértelmű feladat is lehet.

Mondatokra bontás

A szövegeink mondatokra való bontása automatizálható, de nem triviális feladat. Kézenfekvő lenne egyszerűen a mondatvégi írásjeleket nézni, és ezek szerint elszeparálni a mondatokat. Ilyenkor viszont álmondathatárokkal is találkozhatunk, hisz például a pont is előfordulhat mondat közepén, rövidítéseknél, sorszámoknál, dátumokban, vagy akár IP-címekben is. A feladat megoldására általában szabály alapú döntési algoritmust alkalmaznak, mely az ilyen téves mondathatárokra utaló, vagy ezeket cáfoló jelekre és tulajdonságokra tartalmaz szabályokat. Ezeket a szabályokat próbáljuk illeszteni a szövegeinkre. A tulajdonságok lehetnek felszíniek, mint például a jelet követő szó nagy kezdőbetűs e vagy sem, esetleg a jel egy szó közepén van e, stb. Az algoritmus az illesztés mellett kereshet gyakori álmondathatárokat tartalmazó listákban is, mint például a rövidítéslisták. Az ilyen listák nyelvenként eltérőek lehetnek, és időnként a lista karbantartására, frissítésére is szükség lehet. A szabályok tartalmazhatják, hogy a vélt határtól milyen pozícióra elhelyezkedő és milyen jellegű szavakra, vagy pontosabban tokenekre vonatkozik, és szokás megadni hozzájuk egy súly értéket is, hogy amennyiben egy végere több szabvány is illeszkedik a nagyobb súllyal rendelkező juthasson érvényre. Legyen a szabályrendszerűnk bármilyen komplex, és kifinomult is, előfordul, hogy az algoritmus nem képes jól dönteni, mert a mondat, vagy mondatok értelmezésével lehet csak jól meghatározni, a mondatok valódi határát.

Tokenizálás

A token egy bizonyos karaktersorozat konkrét előfordulása, míg típusnak nevezzük az azonos tokenek osztályát. Így a típusok összessége alapján állítható elő a szótár. Ez a tokenizációs lépés minden lényeges szövegbányászati feladatnál, mind a keresés, mind pedig a rendszerezés jellegűeknél, egy fontos előfeldolgozási lépés. Hisz míg például a keresésénél ezekre illesztünk, információkinyerésnél, ezek potenciális információt hordozó entitások lehetnek, addig a dokumentumok rendszerezésénél, ezek olyan attribútumok lehetnek, segítségével mérhetjük az egyes dokumentumok összetartozását, hasonlóságát. Ezek a feldolgozásunk legkisebb részelemei. A gyakorlatban sokszor egyszerű szavak, de mint a fenti definícióból látható lehetnek más szövegegységek, és karaktersorozatok, ritkán és szélsőséges esetekben egész dokumentumok is. A tokenek alatt leggyakrabban mégis elszeparált szavakat értünk. Az ilyen tokenek előállítása elsőre egyszerű is lehet, hisz elég csak az írásjeleket elhagyva, a szóközöknél elválasztani a szöveget. De számos további kérdés is felmerül, mint, hogy hogyan kezelhetjük a kötőjellel elválasztott vagy aposztrófot tartalmazó szavakat, egyben vagy külön tokenként, vagy az egyéb írásjeleket tartalmazó szavakat, mint a kukac és pont karaktert tartalmazó e-mail címeket. Vagy hogyan kezeljük azokat a jeleket, amilyen az informatikusok többségének egyértelműen külön fogalmat reprezentáló C++, itt sem hagyható el a két plusz jel, hisz úgy már egy merőben más fogalmat kapnánk. Kérdés továbbá, hogy miként kezeljük a tulajdon neveket, különböző együttesek vagy csoportok több tagból álló neveit vagy a konkrét eseményt jelölő dátumokat. Természetesen az előbbi kérdésekre több helyes válasz is elképzelhető a feladatunktól függően, illetve a nyelvi és a felhasználási környezet is nagyban befolyásolhatja, hogy konkrétan milyen módon tokenizálunk.

Egy alternatív megoldás is létezik, amit karakter n-gramm módszernek nevezünk. Ekkor a szöveget n hosszúságú karaktersorozatokra bontjuk. Például: a „karakter 4-gramm” szöveget 4-grammokra bontva a következő eredményt kapjuk: kara, arak, rakt, akte, kter, ter , er 4, r 4-, 4-g, 4-gr, -gra, gram, ramm. Ez az eljárás jól használható nyelvek meghatározásakor, vagy más egyszerű osztályozási problémánál, esetleg olyan jellemzően távol-keleti nyelveknél szótárépítésre, ahol nincsenek a különböző szavak szóközökkel elválasztva egymástól. De nem célszerű a használata az európai eredetű nyelveknél olyan feladatoknál, ahol a szótár elemeinek értelmes egységeknek kell lenniük.

A tokenek típusai lesznek a nyelvi elemzés alapvető szövegreprezentáló egységei, és különböző szűrő eljárások után a típusok építik fel magát a szótárat is.

Stoppszószűrés

Egyes szavak, illetve tokenek nagy számban fordulnak elő a dokumentumgyűjtemény dokumentumaiban és nincsenek szoros kapcsolatba azok témájával, ilyenek a névelők, a határozószavak, és a névutók. Ezek a stopszavak. A stopszószűrésre, vagy azok feldolgozás közbeni kiküszöbölésére láttunk egy matematikai jellegű módszert, a td-idf súlyozás szerepében. Tehát a stopszavakat a tokentípusok előállítása után, különböző dokumentumgyakorisági adataik alapján határozzák meg, majd a gyakorlatban esetenként manuálisan ellenőriznek, és ezek egy részét, a feladattól függően elhagyhatják. Napjainkban például keresésénél, esetleg maximum néhány tízes vagy akár tíz alatti szót hagynak el. Míg az elhagyások száma egyes osztályozási és csoportosítási feladatokban ez több százas vagy akár ezres nagyságú is lehet.

Ennek az az oka, hogy a keresésénél, az elhagyott stopszavak szerepelhetnek ismert kifejezésekben, különböző irodalmi vagy más művészeti művek címeiben, és elhagyásukkal ezekre való keresés is nehézkessé válna. Az indexállományok tömörítése, illetve a megnövekedett egyre olcsóbb tárkapacitás együttesen elérték, hogy az idők folyamán egyre kevesebb és kevesebb stopszót kelljen elhagynunk. Mindeközben a rendezési, és más pontos egyezést nem igénylő feladatoknál, ezek a szavak, konkrét jelentés nélkül, csak megnehezítenék a feldolgozást, nélkülük nagy mértékben lecsökkenhet a reprezentációs modell mérete. Gyakran szabják az adott korpuszra, az elhagyható szavak listáját, így biztosítva a lehető leghatékonyabb és mégis biztonságos feldolgozást.

Lemmatizálás és Szótövezés

A stopszószűrés mellett léteznek nyelvészeti eszközök is a szótárak szűrésükre és méretének redukálására. Az alapötlet az, hogy a legtöbb nyelvben vannak a szavaknak különböző módosult alakjai. Ha ezeket a különböző alakokat egy közös alakként tárolnánk, egyes feladatokban elhanyagolható szóalak információ elvesztése mellett, nagy mértékben redukálhatnánk a modell méretét. A csökkentés mértékét befolyásolhatja az adott nyelv morfológiájának gazdagsága. Például ez a mérték az Angol nyelvnél megközelítőleg 40-70 százalék között lehet, addig a Magyarnál elérheti akár a 90%-ot is. A lemmatizálás, és a szótövezés is ilyen közös, úgymond kanonikus alakra hozó módszerek. A szó módosult alakjait a toldalékok, mint a képző, a jel, és a rag, adják. Ezek a toldalékok a szó előtt prefix-ként, és a szó után is állhatnak, körül is vehetik a szótövet, vagy a magyartól idegen módon, be is ékelődhetnek a szótőbe. A közös alakra hozásnál, ezeket a toldalékokat igyekszünk valamilyen módon leválasztani az adott szóról.

A lemmatizálás közben a szó normalizált szótári alakját azaz a lemmáját határozzuk meg. Nyelvészeti motivációjú eljárás közben, mindig értelmes szótári alakot hozunk létre. A lemmatizálás nem egyértelmű, ezért az eredmény több szó is lehet. Például a falunk szó szótári alakja a falu, de a fal szó is.

A szótövezés egyfajta szócsonkolást eredményez. Nem az a célja, hogy értelmes szótári szó jöjjön létre, sokkal inkább az, hogy a kialakított új szó a valódi szövegben is megtalálható legyen. A szótövező eredménye függ az adott algoritmustól.

Hasonlítsuk össze néhány példán a két megközelítést. A munkát és a munkám szavak lemmája a munka, szótöve algoritmustól függően a munka, vagy a munká. A ló lemmája a lovak, lovát és maga a ló szavaknak, de ezek szótöve a választott szótövező függvényében lehet lahet a lo, vagy a ló is.

Információkinyerés

Az információkinyerésnél (Information Extraction - IE) a fő célunk nagy mennyiségű szövegből kigyűjteni a legfontosabb információkat, tesszük ezt olyan formában, hogy azt később akár egy relációs adatbázisba is beírhatjuk. Tehát a strukturálatlan adatokat kívánjuk valamilyen struktúrában összefoglalni. Napjaink meghatározó szövegbányászati kutatási iránya, hisz kiválóan alkalmas lehet nagy mennyiségű emberi munka kiváltására. Az adatok strukturált formába öntésével segíti a folyamatosan növekvő mennyiségű információinkat könnyebben kezelhető, és jobban áttekinthető, jobban ellenőrizhető és feldolgozhatóbbá tenni. Egyik dinamikusan fejlődő altípusa a nyelvközi információkinyerés (Cross-Language IE), melynél az adatokat több nyelvű szövegekből is összeszedjük, és táblázatba öntjük, majd elég csak a táblázatfejléceket lefordítani a kívánt nyelvre.

Az információ-kinyerés nagy mértékben feladatfüggő megoldásokat kíván, mert többnyire csak előre rögzített típusú elemeket vagyunk képesek a szövegekből hatékonyan kinyerni. Fontos lehet ismernünk a felhasználási szakirány egyes jellemzőit, ahhoz, hogy kideríthessük, hogy az adott feladat szempontjából mik a leginkább fontos attribútumok, amiket a szövegből ki szeretnénk gyűjteni, és azokat milyen módon, és formában lehet célszerű a felhasználók számára azokat prezentálni. A megoldásunk továbbá függeni fog az alkalmazási terület jellemző korpuszától is.

Az információkinyerés összehasonlítása más főbb feladattípusokkal

Az információkinyerés a rendezési alaptípusokkal szemben, nem dokumentumokat akar valamilyen szempont szerint besorolni valamilyen kategóriába, mint az osztályozás, és nem célja továbbá dokumentumok valamilyen szempont szerinti csoportosítása sem.

A keresési alaptípusba tartozó információ-visszakereséssel szemben nem lokalizálja a felhasználó számára az információt, nem választja ki a dokumentumkollekció egy kifejezés alapján releváns dokumentumait. Ezzel szemben kiválasztja a feladat szempontjából fontos dokumentumokat, ezeket önmaga elemzi, és a kivont és formába öntött információkat juttatja vissza a felhasználónak. Az információkinyerő rendszerek feladatfüggőek, míg az információ-visszakeresés problémája és megoldása lehet általános. Így a keresőrendszerek kialakításához kevesebb a felhasználási szakterülettel kapcsolatos háttérismeretre van szükségünk. A keresés gyorsabb, és az eredményét a felhasználó manuálisan tekinti át így, ha esetlegesen hibát vét a rendszer annak nincs különösebb kockázata. Ellenben kevesebb munkát is igényel a felhasználótól az információ kinyerő által feldolgozott tömörebb adatmennyiség áttekintése. Az információkinyerő rendszerekben gyakran kell a többi feladattípus megoldásait felhasználni.

Az információkinyerés fontosabb részfeladatai

A probléma megoldása során számos olykor egymásra épülő feladatot kell végrehajtanunk. Ezek rendre a névelem felismerés, a kereszthivatkozás azonosítás, a szereplők azonosítása, a szereplők közti relációk azonosítása, és az események illesztése.

A névelem felismerés

A névelem, vagy angolul Named Entity, egy a valóságban előforduló, névvel ellátott objektum-egyed megnevezése. Az ilyen egyed lehet egy személy tulajdonneve, egy telefonszám, egy e-mail cím, vagy bármilyen más azonosító. Ezek felismerése olykor egyszerű reguláris kifejezésekkel történik, de főleg a tulajdonnév-felismerésnek nevezett alfajában bonyolultabb módszerekre is szükség lehet. A tulajdonnév-felismerés problémakörébe, nem csak a tényleges tulajdonnevek felismerését szoktuk érteni, de különböző együtt kezelendő tokensorozatokat is ide értünk. Például egyes főneveket, mint igazgatóság, vagy bevásárlóközpont esetenként érdemes lehet, a név részeként tekinteni. Manapság a névelem-felismerésre gyakran különböző gépi tanulókat alkalmaznak. Ezek a névelemeket, számukra új, de azért a tanulóadatokhoz hasonló tulajdonságú szövegeken, is eredményesen fedezhetik fel.

További példák lehetnek névelemekre, például hírekben a személyek, helyek, konkrét időpontok, szervezetek, egyes eszközök konkrét megnevezése, de biológiai szövegekben lehetnek akár gének, fajok, fehérjék megnevezései is. A specifikációs szövegekben is előfordulhatnak névelemek, különböző cégek, vagy konkrét személyek formájában, de ennél jellemzőbb, hogy különböző felhasználói szerepköröket kell jelölnünk.

A szereplők azonosítása

A szövegbányászatban szereplők alatt többnyire, bizonyos előre definiált sablonnak megfelelő névelemek előfordulásait értjük. Ilyen szereplők lehetnek a például a vásárlók, termékek, vagy orvosi szövegekben például a betegségek tünetei vagy a gyógyszerek. A specifikációs szövegekben szereplők alatt különböző felhasználói kategóriákat értünk. Ezek a szereplők inkább, mint kategória osztályok jelennek meg a szövegekben, semmint egyedek közös sablonra illeszkedő sokaságaként, ezért a megtalálásukra is némiképp eltérő módot célszerű keresni.

A kereszthivatkozások azonosítása

Ez egy olyan lépés, melyben az azonos egyedre utaló tokensorozatokat és névelemeket rendeljük össze. Ennek egyik válfaja az anaforafeloldás, ami a különböző névmások és utalószavak a megfelelő névelemhez való rendelését jelenti.

A szereplők közti relációk azonosítása

Események illesztése részfeladat

Az információkinyerés során előforduló főbb nyelvészeti problémák jellemzése

Asd

A szabály és a statisztika alapú információkinyerési megközelítések összevetése

Szövegbányászat további főbb feladattípusai

Az alábbi fejezetekben röviden írok a szakdolgozatom szempontjából kevésbé érdekes szövegbányászati feladatokról.

Kereséstámogatás, információ-visszakeresés

Osztályozás

Csoportosítás

Összegzéskészítés, kivonatolás

Szövegelemzés

Webes tartalomkeresés

Válaszkereső rendszerek

Szövegbányászati alapfogalmak

Ontológia

Natural Language Process

Hol lehet fontos a projektemben

Szótövezés – okos szövegkiegészítés.

Információkinyerés – Felhasználói körök, entitások, használati esetek, kiválasztása entitásokhoz esetlegesen attributumok rendelése.

A legismertebb ingyenes szövegbányászati csomagok és rövid bemutatásuk.

StanfordNLP

OpenNLP

GATE

<https://gate.ac.uk/>

UIMA

R nyelv

LingPipe

MagyarLánc

<http://www.predictiveanalyticstoday.com/top-11-free-software-for-text-analysistext-mining-text-analytics/>

<http://stackoverflow.com/questions/15857384/accuracy-annie-vs-stanford-nlp-vs-opennlp-with-uima>

<http://www.jstatsoft.org/v25/i05/paper>

<http://stats.stackexchange.com/questions/7211/what-are-the-text-mining-packages-for-r-and-are-there-other-open-source-text-min>

<http://stackoverflow.com/questions/8695552/algorithm-or-c-sharp-library-for-identifying-keywords-in-a-set-of-messages>

<http://www.codeproject.com/Articles/9461/DotLucene-Full-Text-Search-for-Your-Intranet-or-We>

<http://128.206.119.157/nlp/papers/NLPTools.pdf>

<http://stackoverflow.com/questions/2303098/java-open-source-text-mining-frameworks>

<http://www.searchenginecaffe.com/2007/03/java-open-source-text-mining-and.html>

<http://butleranalytics.com/5-free-text-mining-tools/>