EÖTVÖS LORÁND TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI KAR

Szögi Viktória Rózsa
Alkalmazott matematikus mesterképzés
Számítástudomány szakirány

Szövegfeldolgozás gépi tanulási módszerekkel

Szakdolgozat

Témavezető: Lukács András
Számítógéptudományi Tanszék

Budapest
2018

Köszönetnyilvánítás

Szeretném megköszönni témavezetőmnek, Lukács Andrásnak, hogy támogatott a szakdolgozatom elkészítésében és végig türelemmel és megértéssel kísért a munkám során.

Köszönöm a családom tagjainak, hogy végig mellettem álltak és elviselték a nehezebb időszakaimat. Mindig a lehető legtöbbet nyújtották nekem, amiért nagyon hálás vagyok.

Köszönöm a barátaimnak, hogy kitartottak mellettem, elviselték panaszaimat és mindig bizakodással tekintettek felém.

Végül, de nem utolsó sorban köszönöm oktatóimnak, hallgató társaimnak és munkatársaimnak az átadott tudást és tapasztalatokat.

Tartalomjegyzék

1. Bevezetés

Napjainkban a gépi tanulást és azon belül a mély tanulást használó eljárások olyan alkalmazásokban is előfordulnak, ahol eddig nem voltak jelen. Ahogy a technológia fejlődik, egyre nagyobb teret nyernek maguknak ezek a módszerek. Emellett azt is tapasztaljuk, hogy minél több adatból szereznek ismereteket a modellek, annál jobb eredményeket képesek elérni, ami rendkívül hasznos tulajdonságnak számít, mivel az információ mennyisége az idő múlásával egyre csak nő.

A gépi tanulás megítélése jelenleg is nagy ellentéteket szül, de érdemes kiemelni, hogy eddig sok területen segíti és könnyíti az emberek életét. Rengeteg olyan alkalmazás van, ami a hátrányos helyzetű személyeket próbálja támogatni, ilyen például a kerekesszék szemmel való irányítása [31] vagy felirat készítés a jelnyelvhez [38]. Azonban számos olyan is létezik, amelyeket szinte mindenki használ a hétköznapi élete során: levélszemét szűrőket [42, 18], arcfelismerőt fénykép készítésnél [36] vagy akár az ajánlórendszereket [44]. Sok törekvés van például arra is, hogy valós idejű fordító programot tudjunk készíteni, hogy ezáltal áthidaljuk a nyelvi akadályokat a különböző kultúrák között. Véleményem szerint az egyik célja a gépi tanulásnak az is, hogy ne nekünk kelljen a „gépek nyelvére” lefordítani, hogy milyen feladatot várunk el tőle, hanem ő értse meg a mi szavainkat.

A gépi tanulásnak egy fontos kutatási és felhasználási területe a természetes nyelvfeldolgozás. Nem véletlenül van kiemelt szerepe, hiszen a nyelv az elsődleges kifejezési formánk, legtöbbször a nyelv használatával adjuk mások tudtára, amit közölni szeretnénk. A nyelvi modellezést többféle szempontból is vizsgálhatjuk. Elemezhetjük szemantika vagy szintaktikai szemszögből is, de akár modellezhetjük feladatok alapján is, amiket meg szeretnénk oldani. A nyelvfeldolgozás egyik nehézsége, hogy végtelen sok különböző mondatot alkothatunk. A nyelvi gazdagság miatt a szabályalapú megközelítések erősen korlátozottak és ebből adódóan is érdemes a nyelvekből gépi tanulási módszerekkel igyekezni kinyerni a szükséges kapcsolatokat.

A szakdolgozatban először egy rövid áttekintést nyújtunk a gépi tanulási módszerekről, különleges tekintettel a mély tanulási eljárásokra. Ez a felvezetés szükséges ahhoz, hogy jobban megértsük a később taglalt nyelvi modellezési feladatok működését. Végül egy szemléletes és érdekes feladattal igyekszünk az olvasónak figyelmét felkelteni a témakör nyújtotta lehetőségekre.

2. Gépi tanulás és mély tanulás

Jelen fejezetben a gépi tanulást (*machine learning*) és azon belül a mélytanulást (*deep learning*) mutatjuk be, annak érdekében, hogy a később tárgyalt modellek működéséről átfogó képet kapjunk. A fogalmak ismertetése Goodfellow, Bengio és Courville Deep learning című könyve alapján kerülnek bemutatásra [11].

2.1. Gépi tanulás

A gépi tanulás tekinthető az alkalmazott statisztika egy olyan formájának, ami számítógépek használatával szeretne statisztikai becsléseket adni bonyolult problémákra. A tanulás során azt szeretnénk, ha a számítógép képes lenne a megadott adatokból helyes információkat visszaadni a kitűzött feladatra a megszerzett tapasztalatai alapján. Pontosabban Mitchell definíciója szerint [27]: „Azt mondjuk, hogy egy számítógépes program tanul E tapasztalatból a T feladatok osztályára nézve P teljesítménnyel, ha a teljesítménye a T-beli feladatokon P-ben mérve javul az E tapasztalat megszerzésével."

A gépi tanulás feladatai tipikusan azok, amelyek túl bonyolultak ahhoz, hogy ember által létrehozott szabályalapú programokkal megoldhatóak legyenek. Úgy is tekinthetjük, hogy olyan feladatokat is szeretnénk gépekkel elvégeztetni, amelyekről nem tudjuk, hogy hogyan oldhatóak meg. A feladatokat általában fázisokban írják le, ahol egy fázisban kiderül, hogy hogyan kellene egy példát feldolgozni gépi tanulási algoritmussal. Részletesebben kifejtve, egy példa sajátosságok (*features*) egy gyűjteménye, amelyeket mennyiségileg mérünk egy tárgyról vagy eseményről, és ezt szeretnénk gépi tanulási rendszerrel feldolgozni. Például, ha a feladat, hogy szeretnénk arcfelismerésre megtanítani egy gépet, akkor példának tekintünk egy képet, és a sajátossága lehet egy pixelének az értékei. A tanulás során a pixelek tulajdonságait használjuk fel ahhoz, hogy a kép megfelelő részeit be tudjuk azonosítani.

Mivel szeretnénk kiértékelni, hogy a gépi tanulási algoritmusoknak milyen a teljesítménye, szükségünk van hozzá egy mennyiségi mértékre. Általában ez a P teljesítményi mérték függ a rendszer által elvégzett T feladattól is. Például a klasszifikációnál gyakran pontosságot (*accuracy*) használunk arra, hogy megmérjük mennyire jó a modellünk, azaz a helyes eredményt adó adatok arányát tekintjük. Azonban tekinthetjük a hibaarányt is (*error rate*), ami a rosszul megbecsült adatok hányadát nézi.

Aszerint, hogy milyen tapasztalatok megengedettek a tanulás során két nagy csoportba is sorolhatjuk a gépi tanulási algoritmusokat: felügyelt és felügyelet nélküli tanulásra. A felügyelt tanulás esetén minden példát egy címkével vagy

céltárggyal szeretnénk társítani, azaz ezekben az esetekben ismert a helyes eredmény, amit elvárunk a folyamat kimeneteként. Az elvárt értéket szokás célként vagy célkimenetként is nevezni, és a tanulás során a kapott kimenet és a célkimenet közötti különbség minimalizálása a cél. A klasszifikáció is a felügyelt tanulás közé tartozik, előre meghatározott diszkrét kategóriákba szeretnénk osztani a példákat. A felügyelet nélküli algoritmus a hasznos tulajdonságait próbálja megtanulni az adott adathalmaz struktúrájának. A modell eredményének sikerességét gyakran egy kapcsolódó költségfüggvény segítségével mérik, aszerint, hogy ezt mennyire képes az eljárás csökkenteni vagy növelni. Ilyen például a klaszterezés, ami az adatokat különböző csoportokba osztja az osztályozandó pontok hasonlósága szerint.

A felügyelet nélküli tanulásra úgy is tekinthetünk, hogy megfigyeli számos példáját egy x vektornak és megpróbálja megtanulni a $p(x)$ valószínűségi eloszlását vagy valamilyen számunkra érdekes tulajdonságát ennek az eloszlásnak. Ezzel szemben a felügyelt tanulás a véletlen x vektor példái mellett egy hozzá tartozó y értéket is megfigyel és megpróbálja megjósolni y-t x-ből, általában a $p(y \mid x)$ meghatározásával. Tehát a felügyelt esetben y előre meg van mondva a gépi tanulási rendszernek egyfajta útmutatásképp, és arra vonatkozóan kell tanulnia. Fontos megjegyezni, hogy nem élesen elválasztható a két fogalom egymástól, rengeteg algoritmus megoldható mindkét módszerrel. Ha tekintjük a $p(y \mid x)$ felügyelt tanulási problémát, könnyen átalakíthatjuk felügyelet nélkülivé:

$$p(y \mid x) = \frac{p(x, y)}{\sum_{y'} p(x, y')} .$$

Azonban egy felügyelet nélküli feladat is sokszor átfogalmazható felügyelt tanításúvá, például, ha $x \in \mathbb{R}^n$ és használjuk a láncszabályt:

$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1}) .$$

2.2. Mély tanulás

A mély tanulási módszerek a gépi tanulási eljárások közé tartoznak. Mióta a kétezres években megjelent ez az új területe a tanulásnak, ezen technológia fejlődésével többféle definíció is megfogalmazódott róla. Tekinthetünk úgy rá, mint a gépi tanulás egy olyan osztálya, amely nemlineáris információ feldolgozást használ több rétegen keresztül – felügyelt vagy felügyelet nélküli módon – sajátosságok kiértékelésére, transzformációkra, minta felismerésre és klasszifikációra. Azt is mondhatjuk, hogy olyan algoritmusok amik több, különböző absztrakciós szintnek megfelelően próbálnak tanulni. Két fontos tényezőt érdemes kiemelni,

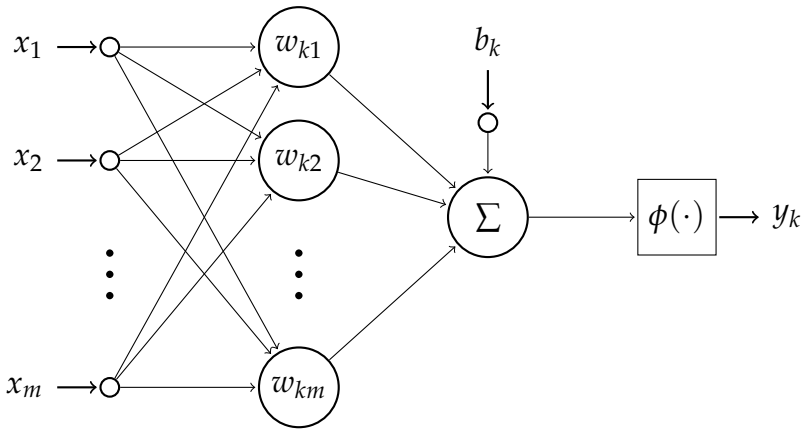
ha a mély tanulási eljárásokról beszélünk, hogy a modellek több rétegű – innen ered a mély elnevezés – nemlineáris információ feldolgozó módszerből állnak, illetve a sajátosságok reprezentációjához egymás után egyre absztraktabb rétegeket használnak. Ezeknek a statisztikai modelleknek a magasabb szintjei alacsonyabb szintű elgondolásokból állnak elő, ám az alacsony szintű megoldások általában nem csak egyfajta magasabb szintű fogalom meghatározására képes, hanem többfélére is. A manapság leginkább használt mély tanulási modellek alapjai a neurális hálózatok (*neural network*), ezért először rövid áttekintést nyújtunk róluk, majd láthatunk néhány speciális hálózat típust is [8].

Egy neurális hálózat egy erőteljesen párhuzamosan elosztott processzor, amely egyszerű feldolgozó egységekből épül fel, amelyeknek természetes hajlamuk a tapasztalatokból szerzett tudás tárolása és hozzáférhetővé tétele. A neurális hálózatokat a biológiai idegrendszer működéséről mintázták és két szempontból hasonlítanak az agyra:

- a tudást a hálózat tanulás során a környezetből szerzi meg,
- a súlyok tárolják a megszerzett tudást.

Más szavakkal élve a mesterséges neurális hálózatok az emberi agy modellezési fajtái, ahol a feldolgozó egységekre neuronokként tekintünk. Egy neuron modellezésénél a következő három elem fontos szerepet játszik:

1. A szinapszisok vagy összekötő láncok egy halmaza. Ezeknek mindegyikéhez tartozik egy-egy súly. Formálisabban, ha a j összekötő lánc bemenete az x_j jel, és a lánc a k neuronhoz kapcsolódik, akkor a jel a w_{kj} súllyal lesz megszorozva. Az agytól eltérően itt a súlyok lehetnek negatív értékek is.
2. Egy összeadó, ami összegzi a bemeneti jeleket a megfelelő súlyokkal, azaz vesz egy lineáris kombinációt.



1. ábra. Neuron modell

3. Egy aktivációs függvény, amely a kimenet nemlinearitását tudja adni....

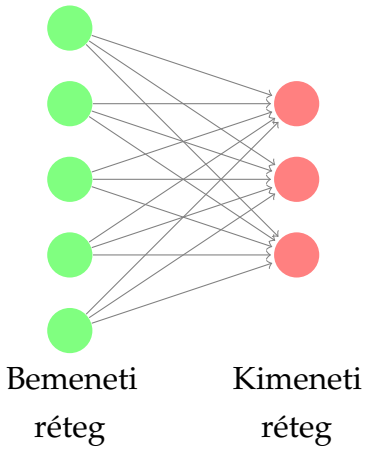
A neurális modellek gyakran tartalmaznak egy torzítást/eltolásértéket (*bias*) is, jelöljük ezt most b_k-val. Az eltolásérték az aktivációs függvény bemeneti értékét befolyásolja. Most formálisan is leírjuk a modellt és az 1. ábrával szemléltetjük:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

$$y_k = \phi(u_k + b_k)$$

A fenti egyenletekben $x_1, \ldots, x_m$ a bemeneti jelek, $w_{k1}, \ldots, w_{km}$ a hozzájuk tartozó súlyai a k neuronnak, u_k a lineáris kombináció kimenete a bemenetre vonatkozóan, b_k az eltolásérték, $\phi(\cdot)$ az aktivációs függvény és y_k a neuron kimeneti jele [16].

2.2.1. Egyrétegű előrecsatolt neurális hálózat



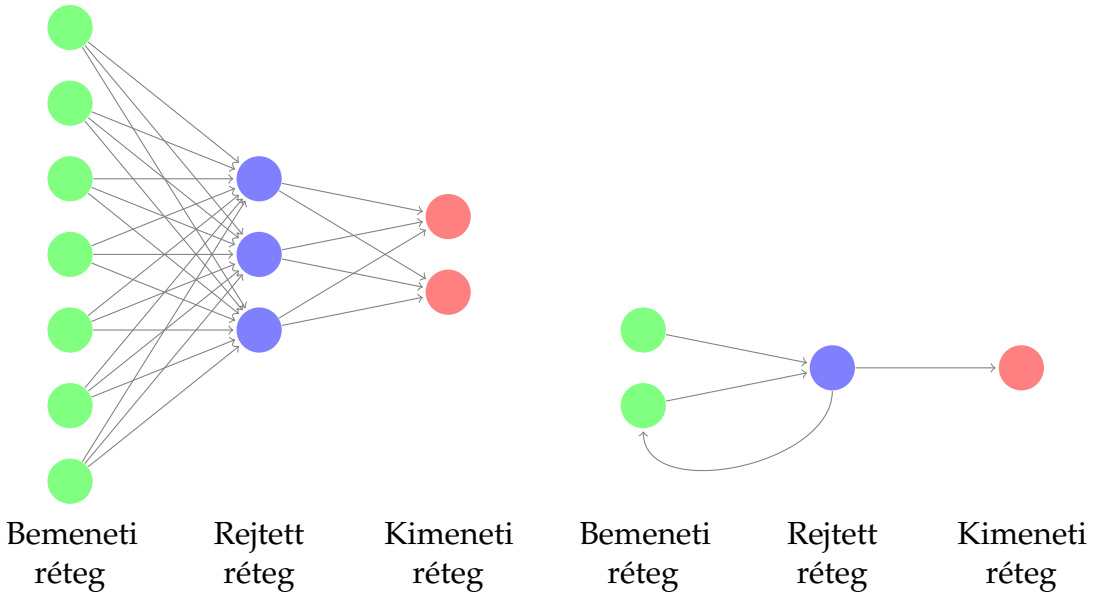
2. ábra

A neuron modell után, megjelentek a továbbfejlesztett algoritmusok, amelyek arra épülnek, hogy több rétegbe rendeznek neuronokat és ezeket összekapcsolják. A legegyszerűbb rétegezett neurális hálózat az egyrétegű előrecsatolt hálózat (*single-layer feedforward network*), amiben van egy bemeneti rétege (*input layer*) a forrás csúcsoknak és ezek közvetlenül kapcsolódnak a kimeneti (*output*) réteg neuronjaihoz. Azonban a kimeneti réteg csúcsai nem csatlakoznak a bemeneti réteghez (lásd a 2. ábrát), emiatt hívjuk előrecsatolt hálózatnak. A bemeneti csúcsokon nem végzünk számításokat, ezért ezt a réteget nem vesszük figyelembe a hálózat névadását illetően, így egyrétegű hálónak nevezzük ezt a modellt.

2.2.2. Többrétegű előrecsatolt neurális hálózat

A többrétegű (*multilayer*) előrecsatolt hálózat (3a. ábra) abban különbözik az egyrétegűtől, hogy a bemeneti és kimeneti rétegen kívül tartalmaz még egy vagy több rejtett (*hidden*) réteget. Ezekben a rejtett rétegekben is számítási csúcsok vannak, és rejtett neuronokként vagy rejtett egységekként hivatkozunk rájuk.

A rejtett elnevezés onnan ered, hogy a hálózat ezen részeit nem látjuk közvetlenül a tanulás során. A rejtett rétegek hozzáadásával a hálózat képes lehet magasabb rendű statisztikákat kinyerni a bemenetből. Tekinthetünk rá úgy, hogy a háló globálisabb képet nyer az adatokról a több kapcsolat miatt. A hálózatban a bemeneti réteg csúcsai az első rejtett réteg megfelelő neuronjaihoz kapcsolódnak, míg a rejtett réteg neuronjai mindig a következő rejtett réteghez kapcsolódnak. Legvégül az utolsó rejtett réteg a kimeneti réteghez kapcsolódik. A kimeneti rétegből származó kimeneti jeleket tekintjük a hálózat válaszának a bemenetre vonatkozóan.



(a) Többrétegű előrecsatolt neurális hálózat (b) Visszacsatolt neurális hálózat

2.2.3. Visszacsatolt neurális hálózatok

A visszacsatolt (*recurrent*) hálózat abban tér el az előrecsatolt társától, hogy van benne visszacsatolt hurok, azaz van olyan kapcsolat a hálóban, ami egy későbbi rétegből egy korábbiba vezet. Ezeket a visszacsatolásokat egyfajta memóriaként használja a hálózat a tanítás során. Ha több tanítási fázis van, akkor például a t-edik fázisban felhasználhatjuk a $t-1$-edikben kiszámított információinkat.

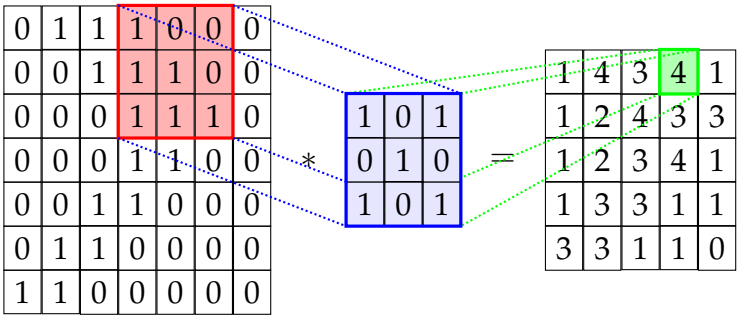
2.2.4. Konvolúciós neurális hálózatok

A konvolúciós neurális hálózatok (*convolutional neural networks*) olyan neurális hálózatok, amelyeket képeken való mintafelismeréshez fejlesztettek ki. A motiváció a modell mögött, hogy a korábbi eljárásokkal nagyon sok neuronra volt szükség már az első rejtett rétegben, ha a bemeneti réteg egységeiként a kép pixeleit képzeljük el. Egyszerűen a rétegek számának növelésével nem lehet áthidalni

a nagy dimenzió problémáját. Ennek az egyik oka a korlátozott számítási képesség, hiszen minél összetettebb egy hálózat, a tanítás is annál költségesebb lesz. A másik ok a túltanulás (*overfitting*), amikor ahelyett, hogy az általános összefüggéseket tanulna meg a modell, a minta egyedi sajátosságait jegyzi meg [21, 29].

A konvolúciós hálókban az egy rétegben levő neuronok három dimenzióba vannak rendezve, és ha a képekre gondolunk, az első kettő a geometriára vonatkozik (szélesség, magasság), a harmadik a mélység, alapvetően ebből nyerjük majd ki a sajátosságokat . Emellett jellemző ezekre a hálózatokra, hogy egy adott rétegben levő neuronok, csak a megelőző rétegbeli neuronok egy részéhez kapcsolódnak. Ha például van egy 28 pixel széles és 28 pixel magas színes képünk, akkor a bemenet $28 \times 28 \times 3$ dimenziós lesz, ahol a mélység tengelyén a pixelek sajátosságait szeretnénk kinyerni és mivel színes képről van szó egy pixel megjelenése jól leírható a piros, zöld és kék színhármassal, emiatt lesz a három a mélység értéke. A kimenet $1 \times 1 \times n$-es lesz, ahol n jelölheti az osztályok számát, amikbe a képeket be szeretnénk kategorizálni. A konvolúciós hálók háromféle rétegből állnak:

- **konvolúciós réteg:** itt tanítható magokat vagy más néven szűrőket (*kernel*) használunk. Ezek a magok általában alacsony térbeli dimenzióval rendelkeznek, de megőrzik a bemenet mélységének dimenzióját. A tanítás során



4. ábra. Konvolúció[1]

ebben a rétegben minden szűrőt – lásd a 2. mátrixot a 4. ábrán – végigcsúsztatunk a bemeneten – 1. mátrix a 4. ábrán – széltében és hosszában is és kiszámoljuk a skaláris szorzatokat a szűrő elemei és a bemenet pontjai között. Ennek eredményeképpen létrejön egy-egy 2 dimenziós aktivációs térkép – 4. ábra 3. mátrixa –. Három hiperparamétert is meg kell adnunk ennél a rétegnél. Az első a szűrők száma, ugyanis ez megegyezik a réteg kimenetének a mélységével. A következő a lépésköz, hogy hány egységgel

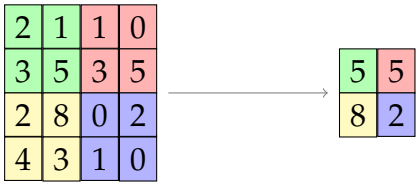
[1] Az ábra a következő oldal alapján készült:
https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

csúsztatjuk odébb a szűrőt a konvolúció során – ennek értéke általában 1 szokott lenni –. A harmadik paraméter a nulla-feltöltés (*zero-padding*), ami azt határozza meg, hogy a bemeneti mintát hány réteg nulla értékű mezővel határoljuk körül (lásd az 5. ábrát). A bővítés lényege, hogy ezzel szabályozni tudjuk a kimenet dimenzióját (a szélességet és a magasságot), így akár a bemenettel megegyező dimenziós értéket kaphatunk.

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	0
0	0	1	0	1	1	1	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	1	0	1	0	0
0	0	1	0	1	1	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

5. ábra. Nulla-feltöltés

- **összevonó** (*pooling*) **réteg:** a célja a dimenziócsökkentés, aminek hatására a számítási összetettség kisebb lesz és a túltanulást is korlátozza. Az eljárás, hogy egy ablakkal (ami általában egy 2×2-es mátrix) végigmegyünk az aktivációs térképen, és vesszük az elemek maximumát, vagy az átlagát. Általában 2-es lépésközzel haladunk. Minden aktivációs térképre külön hajtjuk végre.



6. ábra. Maximum összevonó

- **teljesen összekötött** (*fully-connected*) **réteg:** ezen a szinten az összes neuron közvetlenül kapcsolódik a két szomszédos réteg összes neuronjához, de rétegen belül nincsenek összeköttetésben.

3. Nyelvi modellezés

A természetes nyelvfeldolgozás (*natural language processing*) mindennapjaink során is fontos szerepet tölt be. Vannak például olyan eljárások, amelyek a helyesírásunkat ellenőrzik, ezek segítségünkre lehetnek levélírás során is. Emellett az automatikus kiegészítő eljárások is meggyorsíthatják a rutin teendőinket. Például amikor internetes keresőrendszert használunk, gyakran nem kell kiírnunk a teljes kifejezést, a szolgáltatás már egy-két szó után felajánl lehetőségeket, hogy miről listázzon nekünk találatokat. Az üzeneteink írását is segíthetik ezek az eljárások, amik pár karakter beírása után kiegészítik a leírni kívánt szavainkat vagy kijavítják a tévesen bevitt szövegünket. Mivel még rengeteg felhasználási területe van a nyelvfeldolgozásnak, ezért a dolgozatban nem térünk ki az összesre, ámbár ismertetünk néhány eljárást ebben a fejezetben, amelyek mély tanuláson alapulnak.

3.1. Szöveg klasszifikáció

A szöveg klasszifikáció lényege, hogy a vizsgált dokumentumhoz hozzárendeljen egy címkét, ezzel besorolva őt egy adott témába. A kategóriák, amikbe besorolható a szöveg, előre meg vannak adva. Az alább felsorolt feladatok is ebbe a csoportba tartoznak.

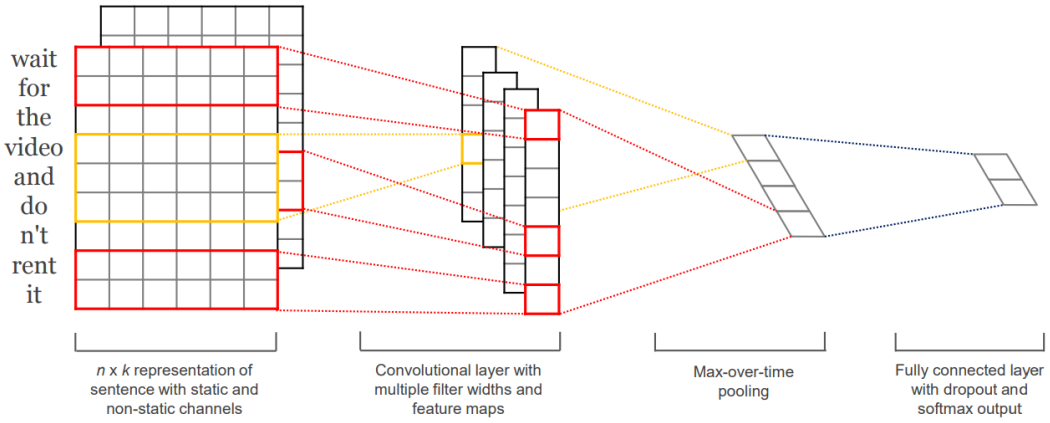
- **Érzelem elemzés** (*sentiment analysis*)**:** a folyamat során azt szeretnénk eldöntetni, hogy a vizsgált szöveg negatív vagy pozitív hangvételű; esetleg egy kicsit finomabb skálán próbáljuk meg eldönteni, hogy milyen hangulatú a vizsgált bemenet. Gyakran használják ezt a módszert termékek értékelésének elemzésekor, vagy akár filmekről írt vélemények besorolására.

- **Levélszemét szűrés:** érdemes említést tenni a spam szűrő rendszerekről (*spam detection*) is, hiszen ez az eljárás fontos szerepet tölt be abban, hogy az egyszerű felhasználók ne essenek áldozatul hamis üzeneteknek. Emellett pedig kellemesebbé is teszi a levelező rendszerek használatát, hiszen sok nem kívánatos, érdektelen üzenetet szelektál ki, így hatékonyabbá teszi a fontos információkhoz való hozzáférést.

- **Nyelv felismerés:** Meghatározza, hogy milyen nyelvhez tartozik a dokumentum. Olyankor lehet fontos, ha meg szeretnénk érteni egy külföldi irodalmat, de nem tudjuk, hogy milyen eredetű.

- **Műfaj felismerés:** A dokumentumokat klasszifikálja műfajok szerint.

A szöveg klasszifikáció során gyakran alkalmaznak konvolúciós neurális hálókat [41, 22, 19, 20], ezért most példaként röviden bemutatjuk Yoon Kim mondat klasszifikációs modelljét [22]. Ez az eljárás olyan konvolúciós hálózatokat használ, melynek bemenetei szóvektorok. Ezeket a szóvektorokat egy másik nyelvi modellből veszi, amikről részletesebben beszélünk a 4. fejezetben. Jelölje $x_i \in \mathbb{R}^k$ a mondat i-edik szavához tartozó k dimenziós szóvektort, amit a másik nyelvi modellből nyerünk ki. Ekkor egy n hosszú mondat a következőképpen írható le formálisan: $x_{1:n} = x_1 \oplus x_2 \oplus \ldots \oplus x_n$, ahol $\oplus$ az összefűzés (*concatenation*) műveletét jelöli. Bevezetünk egy általánosabb jelölést is: $x_{i:i+j} = x_i \oplus x_{i+1} \oplus \ldots \oplus x_{i+j}$, ami egymás utáni j szó összekapcsolását adja meg. A konvolúciós rétegben egy $w \in \mathbb{R}^{h \times k}$ szűrőt használunk, amely h szóra alkalmazva eredményez egy új sajátosság vektort. Tehát, ha az $x_{i:i+h-1}$ szavakra alkalmaztuk a szűrőt, akkor a $c_i = f(w \cdot x_{i+h-1} + b)$-t kapjuk, ahol b az eltolásértéket jelöli és f egy nemlineáris függvény. A szűrőt végigcsúsztatva a mondat szavain kinyerjük a $c = [c_1, c_2, \ldots, c_{n-h+1}]$ aktivációs térképet. Az összevonó rétegben max-overtime [7] függvényt használunk, ami megadja a legnagyobb értékű, azaz a legfontosabb sajátosság vektort az aktivációs térképre vonatkozóan: $\tilde{c} = \max c$. Ennek az összevonó technikának nem okoz gondot, hogy változó hosszúságú mondataink vannak. A modellben több különböző szűrőt is használnak, eltérő ablak méretekkel, hogy ezáltal többféle sajátosságot kapjanak. Az összevonó réteg kimeneteként kapott egységekre egy teljesen összekötött szoftmax réteget alkalmazunk. Végül ennek az eredménye egy eloszlást ad a címkék felett. A teljes folyamat megtekinthető a 7. ábrán.



7. ábra. Konvolúciós hálózat egy példamondatra

3.2. Beszédfelismerés

A beszédfelismerő eljárások célja, hogy a gépek megértsék amit az emberek mondanak. Általában az elhangzott beszédet írásos szöveggé alakítják ezek a módszerek, majd valahogyan értelmezik és felhasználják a hangokból kinyert tartalmat. Ilyen feladatok például a beszéd vagy a filmek feliratozása, de lehet egy olyan rendszer, amely vezetés közben képes irányítást adni egy megadott kérdésre vonatkozóan. A rekurrens és a konvolúciós neurális hálók alkalmazása gyakori ebben a feladatkörben [45, 1, 14, 15]. Az alábbiakban ismertetjük a [45]. publikációban bemutatott modellt.

A következő modellt arra tesztelték, hogy képes-e a megfelelő fonémákat társítani a hangokhoz. Ez az eljárás egy hibrid felépítésű módszer, mivel két technikát ötvöz, ugyanis konvolúciós neurális hálót és kapcsolatos időzített klasszifikációt (*connectionist temporal classification*) is használ. A korábbi módszerek konvolúciós hálózatok helyett általában visszacsatolt hálózatokkal dolgoznak, mivel jó eredmények érhetőek el velük a beszédfelismerés során. Azonban ezeknek a hátránya, hogy nagyon lassan tanulnak és a gradiens értékek „felrobbanhatnak" a tanulás során, mivel ezek egyre nagyobb értékeket vesznek fel [5]. Az említett problémáknak a kiküszöbölésére szolgál az újabb modell. Ahhoz, hogy az eljárás eredményesen működjön, szükség van hosszútávú függőségek használatára, emiatt több egymásba ágyazott konvolúciós réteget tartalmaz a modell. Emellett kisméretű szűrővel dolgozik, ami a lokális sajátosságok megtanulását segíti elő. A rétegek a következőképpen követik egymást:

- Bemeneti réteg.
- Konvolúciós réteg, amit egy maxout aktivációs függvény követ.
- Maximum-összevonó réteg.
- Újabb konvolúciós rétegek, amiket szintén maxout követ. Ez kilencszer ismétlődik meg egymás után.
- Három teljesen összekötött réteg, amiket megint csak maxout aktivációs függvény követ.
- Kapcsolatos időzített klasszifikáció.
- Kimenet.

Konvolúció: Jelölje $X \in \mathbb{R}^{c \times b \times f}$ a bemenetként adott akusztikus értéket, ahol c a csatornák száma, b a frekvencia sávszélessége és f az időtartam. A rétegben k darab szűrő van, ahol minden egyes $W_i \in \mathbb{R}^{c \times m \times n}$. A $H_i \in \mathbb{R}^{k \times b_{H_i} \times f_{H_i}}$ aktivációs függvényeket a következőképpen kapjuk meg: $H_i = W_i * X + b_i$, ahol

b_i egy eltolásérték, $*$ pedig a konvolúciós műveletet jelöli. A modell nulla-feltöltést használ, hogy az eredményül kapott f_{H_i} dimenziók megegyezzenek f-fel. A konvolúciós lépésköz értéke 1.

Maxout: A maxout aktivációs függvény két aktivációs térkép jelöltnek veszi a maximumát: $\tilde{H}_i = \max(H_i', H_i'') = (W_i' * X + b_i', W_i'' * X + b_i'')$ [12].

Összevonó: Az összevonást csak a frekvencia tengely mentén végezzük el, az idő tengely mentén nem. Egy maximum összevonót használunk, ahol p az összevonó mérete és s a lépéstávolság. Formálisan az (r, t) pozícióban kapott kimenete az összevonó rétegnek: $[\hat{H}_i]_{r,t} = \max_{j=1}^{p} [\tilde{H}_i]_{r \times s + j, t}$. A modellben csak egy ilyen réteget használunk, mivel több összevonó használatával csökkenhet a hatékonysága annak, hogy a különböző sajátosságokat megtanulja a rendszer [34].

Kapcsolatos időzített klasszifikáció: Tekintsük azt a feladatot, hogy az $X = \{x_1, \ldots, x_T\}$ bemeneti szekvenciából szeretnénk megkapni $Z = \{z_1, \ldots, z_L\}$ cél szekvenciákat, azaz beszédfelismerésnél az X akusztikus jelből szeretnénk a Z szimbólumok szekvenciáját megkapni. Tehát a $p(Z \mid X)$ valószínűséget szeretnénk minden bemenet-kimenet párra maximalizálni. Az egyik módszer ahhoz, hogy megkapjuk a változó hosszúságú kimeneti szekvenciák eloszlását a náluk sokkal hosszabb szekvenciájú bemenetekből, az egyik mód, hogy bevezetünk $O = \{o_1, \ldots, o_T\}$ látens szekvenciákat. Majd egyes Z szekvenciák valószínűségét, a hozzá tartozó látens szekvenciák összegéből kapjuk meg. A klasszifikáció minden időlépésben meghatároz a hálózat kimenetéhez egy eloszlást a látens szekvenciák felett a szoftmax függvény segítségével. Ez a kimeneti jelek ábécéjének címkéihez $p(o_t \mid X)$ valószínűséget köt az adott időben. Egy „üres" címkét is használunk, ami annak a valószínűségét reprezentálja, hogy egy szimbólumnak az adott lépésben nem volt megfelelő kimenete. Az így kapott eloszlást átalakítjuk egy olyan σ leképezést meghívva egymás után több alkalommal, amely először egyesíti az egymást követő nem üres címkéket egy címkévé, majd kitörli az üres címkéket. Formálisan leírva a következő a célunk:

$$h(x) = \arg\max p(Z \mid X) = \arg\max \sum_{o \in \sigma^{-1}(Z)} p(o \mid X).$$

A modell a maximalizálás megoldására a legjobb útvonal dekódolást használja (*best path decoding*, [14]).

3.3. Gépi fordítás

A gépi fordítás a különböző nyelvekből adódó problémákat szeretné áthidalni azáltal, hogy szöveget vagy beszédet fordít egy adott nyelvről egy másik nyelvre. Különböző neurális háló modelleket is kifejlesztettek a fordító programok

fejlesztésére [23, 24, 3, 37], mi most egy olyat tekintünk példaként, ami visszacsatolt neurális háló használatára épül és egy úgynevezett kódoló-dekódoló (*encoder-decoder*) eljárást használ [4]. A kódoló eljárás általában egy fix hosszúságú vektorrá alakítja a forrás mondatot, majd a dekódoló kimenete egy fordítást ad a kódolt mondatról. A kódoló és a dekódoló folyamatok együtt maximalizálják a fordítás eredményességének valószínűségét.

Először tekintsük a legegyszerűbb visszacsatolt neurális hálóhoz tartozó kódoló-dekódolót. A bemeneti mondatot jelölje $x = (x_1, x_2, \ldots, x_{T_x})$, amit a kódoló c vektorba olvas be. A rejtett réteg állapota t időpontban $h_t = f(x_t, h_{t-1})$, ahol f egy nemlineáris függvény. c vektort pedig a rejtett réteg állapotaiból generáljuk egy q nemlineáris függvénnyel: $c = q(\{h_1, \ldots, h_{T_x}\})$. A dekódolót gyakran arra tanítjuk, hogy megjósolja az $y_{t'}$ szót a c vektor és a korábbi megjósolt $y_1, \ldots, y_{t'-1}$ szavakból:

$$p(y) = \prod_{t=1}^{T_y} p(y_t \mid \{y_1, \ldots, y_{t-1}\}, c),$$

ahol $y = (y_1, \ldots, y_{T_y})$. Felhasználva a visszacsatolt háló sajátosságait a feltételes valószínűséget a következő alakban is felírhatjuk, ahol s_t a rejtett réteg egy állapotát jelöli és g nemlineáris, többrétegű függvény:

$$p(y_t \mid \{y_1, \ldots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c).$$

Most tekintsük egy fejlesztett változatát a korábbi modellnek, ahol egyszerű visszacsatolt háló helyett kétirányú visszacsatolt hálót [35] használunk kódolóként. Először taglaljuk a dekódoló felépítését. Ebben a modellben a következőképpen felírt feltételes valószínűségeket szeretnénk meghatározni, ahol s_i a háló rejtett rétegének állapotát jelöli az i-edik fázisban:

$$p(y_i \mid \{y_1, \ldots, y_{i-1}\}, x) = g(y_{i-1}, s_i, c_i).$$

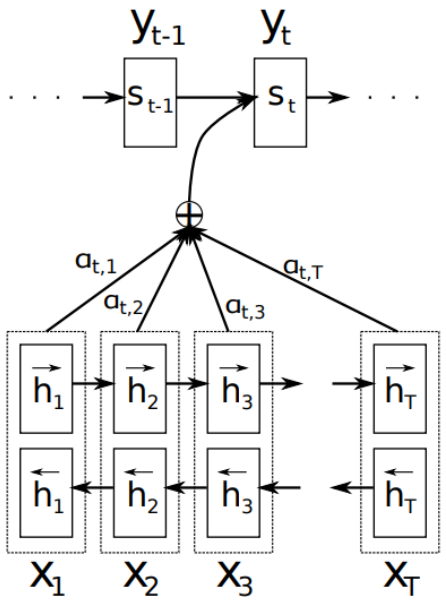
Tehát ebben a modellben különböző c_i vektorokat használunk minden y_i elvárt szóra, a rejtett réteg állapotát pedig a következőképpen írhatjuk le:

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

A c_i vektorok függnek a $(h_1, \ldots, h_{T_x})$ értékektől – amelyekre a kódoló résznél térünk ki részletesebben, mivel ezek a bemenetként kapott mondathoz kapcso-

lódnak – és a következőképpen kapjuk meg:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j, \quad \text{ahol} \quad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad \text{és} \quad e_{ij} = a(s_{i-1}, h_j).$$



8. ábra. A kódoló dekódoló eljárás, miközben $(x_1, x_2, \ldots, x_T)$ bemeneti mondat alapján kiszámolja a t-edikként elvárt szó értékét.

Ezt a modellt, illesztő (*alignment*) modellnek nevezzük, ami azt pontozza, hogy a bemenet a j-edik pozíció körül az i-edik pozíciójú kimenettel mennyire fér össze. a lehet előrecsatolt neurális háló, amely együttesen tanul a rendszer többi részével.

Most tekintsük a kódoló eljárást, hogy teljes képet kapjunk a rendszerről. Az egyszerűbb eljárásban, amikor nem kétirányú hálót használunk,akkor a bemeneti mondatunkat az x_1 szótól kezdve olvassuk be az x_{T_x}-edikig. Vagyis a rendszer, csak a korábbi szavakból használ fel információkat, viszont mi a rákövetkezőket is szeretnénk figyelembe venni. Emiatt használunk kétirányú hálózatot, ami tartalmaz előre irányú és a visszafele irányú visszacsatolt neurális hálót is. Az $\overrightarrow{f}$ előre irányú hálózat x_1-től x_{T_x} dolgozza fel a bemenetet és ezekből kiszámolja az $(\overrightarrow{h_1}, \ldots, \overrightarrow{h_{T_x}})$ rejtett állapotokat. Ezzel szemben az $\overleftarrow{f}$ ellentétes irányú hálózat x_{T_x} -től x_1 fele olvas be, és a $(\overleftarrow{h_1}, \ldots, \overleftarrow{h_{T_x}})$ rejtett állapotokat eredményezi. A korábban említett h_j értékét a két különböző irányú rejtett j-edik állapotok összefűzéséből kapjuk meg: $h_j = [\overrightarrow{h_j^T}; \overleftarrow{h_j^T}]^T$ ami tartalmazza a megelőző és rákövetkező szavak információit is. A modellt a 8. ábra szemlélteti.

3.4. Képszöveg generálás

A folyamat célja, hogy a gép le tudja írni, hogy mi látható egy képen, tehát egy rövid összegzést szeretnénk a képről kapni. Felhasználhatjuk akár videó leírások, ajánlások készítésére is. Ebben a feladatkörben gyakran használnak speciális visszacsatolt neurális hálókat [13, 39, 40].

3.5. Dokumentum összefoglalás

Az eljárás során szeretnénk egy rövid összefoglalást nyerni a megadott tartalomról [6, 33, 28]. Ez lehet egy absztrakt készítése is egy cikkhez, vagy akár egy címadás egy dokumentumnak. Kétféle módszer is lehetséges: a modell használhatja csak a szövegből nyert információkat, vagy korábbi tapasztalatokat is beépíthet a végrehajtás során.

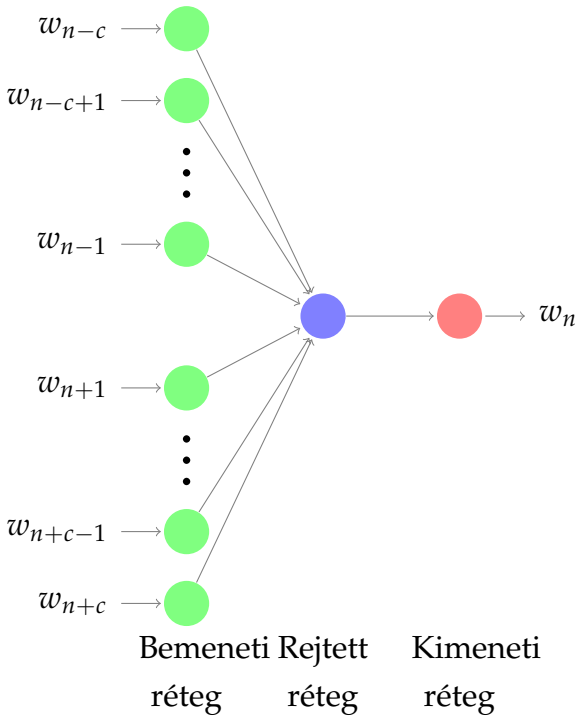
3.6. Kérdés megválaszolás

A kérdés megválaszolás (*question answering*) lényege, hogy egy megadott témakörből kell a gépnek kérdésekre válaszolnia úgy, hogy először feldolgozza a hozzá tartozó irodalmat [17, 43, 9]. Tehát meg kell találnia a kapcsolatot a kérdés és a szöveg között, egyfajta értelmezésről van szó. A kérdésekre általában egyszerű kifejezéseket várnak válaszként például egy nevet vagy egy helyszínt.

4. Vektortér modellek

A vektortér modelleknek a célja, hogy megtanulják a valószínűségi kapcsolatokat, amelyek a szavak között fennállnak. Ezeknek a segítségével lesznek képesek a korábban említett dokumentum összefoglalásra is vagy akár egy mondat kiegészítésére. A továbbiakban három modellt tekintünk részletesen, amelyek nagy népszerűségre tettek szert az elmúlt pár évben, a megjelenésük óta. Ezek a modellek olyan módon reprezentálják alacsonyabb dimenzióban a bemenetként kapott szavakat, hogy közben a szemantikai sajátosságokat megtanulják. Azon szavak, melyek jelentése hasonló, közel kerülnek egymáshoz a kapott sűrű dimenzióban és további tulajdonsága ezen reprezentációnak, hogy a hasonlóan eltérő szavak távolságai közel azonosak, például: $v(\text{anya}) - v(\text{apa}) \approx v(\text{lány}) - v(\text{fiú})$, ahol $v(\text{x})$ az x szó vektor reprezentációját jelöli. Vagyis a távolságok egyfajta szemantikai jelentést foglalnak magukban.

4.1. Continuous Bag-of-Word modell



9. ábra. CBOW modell

A Continuous Bag-of-Word, továbbiakban CBOW modellel azt a folyamatot szeretnénk reprezentálni, amikor egy konkrét szóra következtetünk az ő környezetéből [25, 32, 10]. Másképpen megfogalmazva egy adott szövegkörnyezetet szeretnénk helyesen kiegészíteni a hiányzó szóval. Ennek a szemléltetését láthatjuk az ábrán is. Képletesen, ha $w_1, w_2, \ldots, w_N$-nel jelöljük a teszthalmazunkban adott szavak sorozatát, akkor a következő célfüggvényt szeretnénk maximalizálni a tanítás során:

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{\substack{|t|\leq c\\ t\neq 0}}\log p(w_n \mid w_{n+t}), \qquad (1)$$

ahol c határozza meg, hogy egy szónak mekkora környezetéből szeretnénk jósolni. Például ha $c = 2$, akkor egy szót az előtte levő kettő és az utána levő kettő szavakból szeretnénk kikövetkeztetni. A modellben szoftmax függvény segítségével határozzuk meg a $p(w_n \mid w_{n-c}, w_{n-(c-1)}, \ldots w_{n-1}, w_{n+1}, w_{n+2}, \ldots w_{n+c})$ feltételes

valószínűségeket, ami azt adja meg, hogy mekkora a valószínűsége annak, hogy az w_n-t határoló szavakból helyesen következtetünk rá.

Egyszerűsítés: "bigram" modell

Az egyszerűség kedvéért először tegyük fel, hogy az aktuális szót csak az őt megelőző szóból szeretnénk megjósolni. A modell 3 réteget tartalmaz, egy bemeneti, egy rejtett és egy kimeneti réteget (lásd a 9. ábrán). A modell bemenete egy w_I szó, amit $1 - N$ kódolással adunk meg, ha N a szótárunk mérete. Pontosabban egy olyan (oszlop)vektorként ábrázoljuk a szót, ahol a k-adik elem 1 és minden más 0, ha a w_I k-adik a szótárban. Ha egy D dimenziójú rejtett réteget szeretnénk alkalmazni, akkor egy $N \times D$ méretű W súlymátrixot fogunk használni a bemeneten, formálisan:

$$h = w_I^T W,$$
$$v_{w_I} := w_I^T W.$$

Tehát a használt aktivációs függvény lineáris a rejtett rétegben és egy h kimenetet eredményez. Mivel w_I minden eleme 0 a k-adik kivételével, a súlyozást végrehajtva W-nek k-adik sorát kapjuk meg. Emellett tekinthetjük úgy, hogy W minden sora a bemeneti w_I szó egy vektor reprezentációját adja meg, ezt a fenti egyenletben v_{w_I}-vel definiáljuk.

Ahhoz, hogy a rejtett rétegből meghatározzuk a kimenetet egy másik, $D \times N$ dimenziójú W' súlymátrixot fogunk használni. Jelölje $v'^T_{w_j}$ ennek a súlymátrixnak a j-edik oszlopát. Ekkor a j-edik szót a következőképpen pontozzuk:

$$u_j = v'^T_{w_j} \cdot h.$$

Ezután egy szoftmax osztályozót használva megkapjuk a keresett posterior eloszlást, azaz annak a valószínűségét, hogy a j-edik szót kapjuk, feltéve, hogy a bemenet w_I volt.

$$p(w_j \mid w_I) = y_j = \frac{\exp(u_j)}{\sum_{i=1}^{N} \exp(u_i)},$$

ahol y_j-t a kimeneti réteg j-edik egysége adja meg. Átalakítva az eredményt a bevezetett jelölésekkel:

$$p(w_O \mid w_I) = \frac{\exp(v'^T_{w_O} v_{w_I})}{\sum_{w=1}^{N} \exp(v'^T_w v_{w_I})} \tag{2}$$

Megjegyezzük, hogy v_w-t gyakran szokás a w bemeneti, v'_w-t pedig a w kimeneti vektornak nevezni.

Tanítás

Ebben az esetben a tanítás célja, hogy maximalizáljuk azt a feltételes várható értéket, hogy w_I bemenet mellett, az elvárt w_O lesz a kimenet. Tegyük fel, hogy w_O a szótár i-edik szava. Azaz formálisan:

$$\begin{aligned}\max p(w_O \mid w_I) &= \max y_i = \max \log y_i \\ &= \max \log \frac{\exp(u_i)}{\sum_{k=1}^{N} \exp(u_k)} = u_i - \log \sum_{k=1}^{N} \exp(u_k), \\ -E &:= u_i - \log \sum_{k=1}^{N} \exp(u_k),\end{aligned}$$

ahol $E = -\log p(w_O \mid_I)$ a veszteség függvény, amit szeretnénk minimalizálni.

Backpropagation algoritmust fogunk használni a súlyok javításához. Először a kimeneti és a rejtett réteg közötti súlyokat kell módosítanunk. Ehhez első lépésben ki kell számolnunk minden kimeneti egységre a becslési hibát:

$$\frac{\partial E}{\partial u_j} = y_j - t_j,$$

$$e_j := y_j - t_j,$$

ahol $t_j = 1$, ha a j-edik egység éppen az elvárt kimenet ($j = i$), különben 0.

A W' mátrix hibáját a következő deriváltból kapjuk meg:

$$\frac{\partial E}{\partial w'_{kj}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{kj}} = e_j \cdot h_k,$$

ahol w'_{kj} a mátrix k-adik sorának j-edik elemét jelöli, h_k pedig a h vektor k-adik eleme.

A sztochasztikus gradiens tanulási szabályt alkalmazva kapjuk meg a frissített súlyokat W'-re:

$$w'^{(\text{új})}_{kj} = w'^{(\text{régi})}_{kj} - \eta \cdot e_j \cdot h_k,$$

átalakítva:

$$v'^{(\text{új})}_{w_j} = v'^{(\text{régi})}_{w_j} - \eta \cdot e_j \cdot h \qquad \forall j = 1, 2, \ldots, N, \tag{3}$$

ahol $\eta > 0$ a tanulási ráta.

Következő lépésként a bemeneti és a rejtett réteg közötti súlymátrixot kell javítanunk. Deriváljuk E-t a rejtett réteg kimenetei szerint:

$$\frac{\partial E}{\partial h_k} = \sum_{j=1}^{N} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_k} = e_j \cdot w'_{kj},$$
$$A_k := e_j \cdot w'_{kj}.$$

Itt A_k egy olyan V dimenziós vektor lesz, ami a kimeneti vektorokat súlyozza a becslési hibáikkal, majd ezeket összegzi.

Tekintsük E W szerinti deriváltját:

$$\frac{\partial E}{\partial w_{lk}} = \frac{\partial E}{\partial h_k} \cdot \frac{\partial u_j}{\partial w_{lk}} = A_k \cdot w_{I,l}$$

ahol $w_{I,l}$ a bemeneti vektor l-edik elemét jelöli. Ezzel ekvivalensen felírható a következő formula, ahol $\otimes$ a tenzorszorzatot jelöli:

$$\frac{\partial E}{\partial W} = w_I \otimes A = w_I A^T.$$

Mivel w_I csak egy helyen vesz fel 1-et és mindenhol máshol 0, a fenti egyenlet A^T mátrix egy sorának értékét adja, melynek dimenziója D. Ebből a következő egyenletet kapjuk W frissítéséhez:

$$v'^{(\text{új})}_{w_I} = v'^{(\text{régi})}_{w_I} - \eta A^T.$$

W többi sorát nem kell változtatnunk ebben a lépésben, mivel a deriváltjuk 0.

Kiterjesztés

Az általános modellben nem egy szóból szeretnénk megbecsülni a következőt, hanem egy adott kontextusból szeretnénk meghatározni a hiányzó szót. Az előző egyszerűsített modellt úgy módosíthatjuk ilyen esetben, hogy nem egyszerűen egy bemenetre alkalmazzuk a súlyfüggvényt, hanem az összes bemeneti vektorra. A rejtett réteg bemenetét pedig ezeket átlagolva kapjuk meg:

$$h = \frac{1}{2c} W^T \sum_{\substack{|t| \leq c \\ t \neq 0}} w_t = \frac{1}{2c} \sum_{\substack{|t| \leq c \\ t \neq 0}} v^T_{w_t},$$

ahol c határozza meg, hogy egy szónak mekkora környezetéből szeretnénk jósolni. v_w-t a w bemeneti vektora.

A veszteség függvény a következőképpen határozható meg:

$$E = -\log p(w_O \mid w_{I,-c}, ..., w_{I,c}) = -u_i + \log \sum_{k=1}^{N} \exp(u_k)$$
$$= -v'^T_{w_O} \cdot h + \log \sum_{k=1}^{N} \exp(v'^T_{w_k} \cdot h)$$

A korábbi levezetésekhez hasonlóan megkaphatjuk, hogy most a súlyokat a következőképpen javíthatjuk:

$$v'^{(\text{új})}_{w_j} = v'^{(\text{régi})}_{w_j} - \eta \cdot e_j \cdot h \qquad \forall j = 1, 2, \ldots, N,$$

ahol egyedül a h értéke különbözik az egyszerű modelltől. A bemeneti és rejtett réteg között használt súlymátrixra azt kapjuk:

$$v'^{(\text{új})}_{w_{I,t}} = v'^{(\text{régi})}_{w_{I,t}} - \frac{1}{2c}\eta A^T \qquad \forall |t| \leq c, t \neq 0.$$

Számítási összetettség

Felelevenítjük, hogy a korábbi levezetésünk alapján (kiemelve az 1. és a 2. egyenletet) a modell a következő formában írható fel:

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{\substack{|t| \leq c \\ t \neq 0}} \log \frac{\exp(v'^T_{w_n} v_{w_{n+t}})}{\sum_{w=1}^{N} \exp(v'^T_w v_{w_{n+t}})}.$$

Ennek a modellnek az összetettsége $\mathcal{O}(N \cdot D) + \mathcal{O}(D \cdot N)$, mivel a bemenet $1 - N$ kódolással volt reprezentálva, és a kimeneti vektor dimenziója is N [25, 26]. A legköltségesebb számítás a v' kimeneti vektorok tanítása, hiszen amikor javítani szeretnénk v'_{w_j}-n, akkor ki kell számítanunk a hozzá tartozó e_j becslési hibát (Lásd a 3). egyenletben), amihez a korábbi levezetésben látottak alapján fel kell használnunk az összes szótári szóhoz kapcsolódó w_i súlyt is. Mivel általában hatalmas a szótárunk mérete, beláthatjuk, hogy nem túl hatékony ezt a tanítási eljárást alkalmazni minden fázisban minden egyes v'_j kimeneti vektorra, ezért a 4.4. fejezetben bemutatunk néhány javítási eljárást, amivel a modell komplexitása csökkenthető.

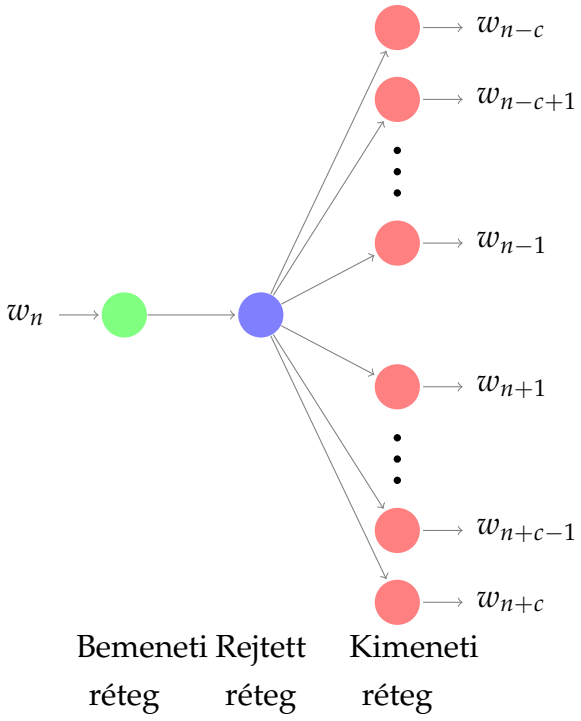
4.2. Skip-gram modell

A Skip-gram modell ismertetését egy feladaton keresztül fogjuk bemutatni. A probléma, amit meg szeretnénk oldani az, hogy egy szóval jellemezzük egy

meghatározott környezetét [25, 26, 32, 10]. Pontosabban egy adott szóból az őt körülvevő szavakra szeretnénk következtetni. Ennek a szemléltetését láthatjuk az ábrán is. Képletesen megfogalmazva, ha $w_1, w_2, \ldots, w_N$-nel jelöljük a tesztalmazban adott szavak sorozatát, akkor a következő célfüggvényt szeretnénk maximalizálni a tanítás során:

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{\substack{|t|\leq c\\ t\neq 0}}\log p(w_{n+t} \mid w_n), \tag{4}$$

ahol c határozza meg, hogy egy szónak mekkora környezetét szeretnénk megjósolni. Például ha $c = 2$, akkor egy szóból az előtte levő két szóra és az utána levő két szóra szeretnénk következtetni. c választásával befolyásolhatjuk az eredmény pontosságát, hiszen ha nagyobbnak választjuk, a tanító halmaz méretét is növeljük, de ebben az esetben a tanítás is több időt fog igénybe venni.



Az alap Skip-gram modellben szoftmax függvénnyel határozzuk meg a $p(w_{n+t} \mid w_n)$ feltételes valószínűségeket, ami azt adja meg, hogy mekkora a valószínűsége annak, hogy az n-edik szóból az $n+t$-edik szóra következtetünk.

Most részletesen ismertetjük azt, hogy mely lépéseken keresztül jutunk el a keresett feltételes valószínűségek meghatározásáig. Amint az ábrán is látjuk a modell 3 réteget tartalmaz, egy bemeneti, egy rejtett és egy kimeneti réteget. A modell bemenete egy w_I szó, amit $1-N$ kódolással adunk meg, ha N a szótárunk mérete. Pontosabban egy olyan (oszlop)vektorként ábrázoljuk a szót, ahol a k-adik elem 1 és minden más 0, ha a w_I k-adik a szótárban. Ha egy D dimenziójú rejtett réteget szeretnénk kapni a bemeneti rétegből, akkor egy $N \times D$ méretű W súlymátrixot fogunk használni a bemenetre, formálisan:

$$h = w_I^T W,$$
$$v_{w_I} := w_I^T W.$$

Tehát egy lineáris aktivációs függvényt használunk a rejtett réteghez, hogy megkapjuk a h kimenetet. Mivel w_I minden eleme 0 a k-adik kivételével, a súlyozást végrehajtva W-nek k-adik sorát kapjuk meg. Emellett tekinthetjük úgy, hogy W minden sora a bemeneti szó egy vektor reprezentációját adja meg, ezért a fenti egyenletben v_{w_I}-vel definiáljuk a w_I ezt a reprezentációt.

Ahhoz, hogy a rejtett rétegből meghatározzuk a kimenetet egy másik, $D \times N$ dimenziójú W' súlymátrixot fogunk használni. Jelölje $v'^T_{w_j}$ ennek a súlymátrixnak a j-edik oszlopát. Ekkor a j-edik szót a következőképpen pontozzuk:

$$u_j = v'^T_{w_j} \cdot h.$$

Mivel $c - c$ darab szót szeretnénk megbecsülni a bemeneti szó előtt és után és mindegyik becsléséhez ugyanazt a súlymátrixot használjuk a rejtett és a kimeneti réteg között, a t-edik helyre a j-edik szóra kapott érték:

$$u_{t,j} = u_j, \qquad \forall |t| \leq c, t \neq 0.$$

Ezután egy szoftmax osztályozót használva megkapjuk a keresett posterior eloszlásokat, azaz annak a valószínűségét, hogy a kapott kimeneti érték megegyezik az elvárt értékkel, feltéve, hogy a bemenet w_I volt.

$$p(w_{t,j} = w_{t,O} \mid w_I) = y_{t,j} = \frac{\exp(u_{t,j})}{\sum_{i=1}^{N} \exp(u_i)}, \qquad \forall |t| \leq c, t \neq 0,$$

ahol $w_{t,j}$ azt jelöli, hogy a j-edik szótárbeli szó szerepel a t helyen a bemenet környezetében. $w_{t,O}$ a kimenetként t-edik helyen elvárt szót jelöli, w_I pedig a bemenetet. $y_{t,j}$-t az utolsó rétegben a t-edik kimeneti egység j-edik eleme határozza meg. Átalakítva az eredményt a bevezetett jelölésekkel:

$$p(w_O \mid w_I) = \frac{\exp(v'^T_{w_O} v_{w_I})}{\sum_{w=1}^{N} \exp(v'^T_w v_{w_I})} \tag{5}$$

Megjegyezzük, hogy v_w-t gyakran szokás a w bemeneti, v'_w-t pedig a w kimeneti vektornak nevezni.

Fontos kiemelni, hogy ilyen módon megvalósítva nagyon költséges a modell, mivel összetettsége a szótár méretének többszöröse. Általában a szótárak hatalmas méretűek, így ez nagy hátránya a modellnek, ezért bemutatunk néhány módszert, amivel hatékonyabbá tehető.

Tanítás

Ebben az esetben a tanítás célja, hogy maximalizáljuk azt a feltételes várható értéket, hogy w_I bemenet mellett, az elvárt w_O lesz a kimenet. Tegyük fel, hogy a szótár i_t-edik szava a t helyen elvárt kimenet a bemenet környezetében.

$$\begin{aligned}\max p(w_{O,-c},...,w_{O,c} \mid w_I) = \max y_i = \max \log \prod_{\substack{|t|\leq c \\ t\neq 0}} \frac{\exp(u_{t,i_t})}{\sum_{k=1}^{N} \exp(u_k)} \\ = \sum_{\substack{|t|\leq c \\ t\neq 0}} u_{i_t} - 2c \cdot \log \sum_{k=1}^{N} \exp(u_k), \\ -E := \sum_{\substack{|t|\leq c \\ t\neq 0}} u_{i_t} - 2c \cdot \log \sum_{k=1}^{N} \exp(u_k).\end{aligned}$$

Backpropagation algoritmust fogunk használni a súlyok javításához. Először a kimeneti és a rejtett réteg közötti súlyokat kell módosítanunk. Ehhez első lépésben ki kell számolnunk minden kimeneti vektor minden egységre a becslési hibát:

$$\begin{aligned}\frac{\partial E}{\partial u_{t,j}} = y_{t,j} - z_{t,j}, \\ e_{t,j} := y_{t,j} - z_{t,j},\end{aligned}$$

ahol $z_{t,j} = 1$, ha a t vektor j-edik egysége éppen az elvárt kimenet ($j = i_t$), különben 0. Az egyszerűség kedvéért vezessük be a következő n dimenziós vektort: $B = B_1, \ldots, B_N$, ahol

$$B_j = \sum_{\substack{|t|\leq c \\ t\neq 0}} e_{t,j}.$$

A W' mátrix hibáját a következő deriváltból kapjuk meg:

$$\frac{\partial E}{\partial w'_{kj}} = \sum_{\substack{|t|\leq c \\ t\neq 0}} \frac{\partial E}{\partial u_{t_j}} \cdot \frac{\partial u_{t_j}}{\partial w'_{kj}} = B_j \cdot h_k,$$

ahol w'_{kj} a mátrix k-adik sorának j-edik elemét jelöli, h_k pedig a h vektor k-adik eleme.

A sztochasztikus gradiens tanulási szabályt alkalmazva kapjuk meg a frissített súlyokat W'-re:

$$w'^{(\text{új})}_{kj} = w'^{(\text{régi})}_{kj} - \eta \cdot B_j \cdot h_k,$$

átalakítva:

$$v'^{(\text{új})}_{w_j} = v'^{(\text{régi})}_{w_j} - \eta \cdot B_j \cdot h \qquad \forall j = 1, 2, \ldots, N,$$

ahol $\eta > 0$ a tanulási ráta.

Következő lépésként a bemeneti és a rejtett réteg közötti súlymátrixot kell javítanunk. Deriváljuk E-t a rejtett réteg kimenetei szerint:

$$\frac{\partial E}{\partial h_k} = \sum_{j=1}^{N} \sum_{\substack{|t| \leq c \\ t \neq 0}} \frac{\partial E}{\partial u_{t,j}} \cdot \frac{\partial u_{t,j}}{\partial h_k} = \sum_{j=1}^{N} B_j \cdot w'_{kj},$$

$$A_k := \sum_{j=1}^{N} B_j \cdot w'_{kj}.$$

Tekintsük E-nek a W szerinti deriváltját:

$$\frac{\partial E}{\partial w_{lk}} = \frac{\partial E}{\partial h_k} \cdot \frac{\partial u_j}{\partial w_{lk}} = A_k \cdot w_{I,l},$$

ahol $w_{I,l}$ a bemeneti vektor l-edik elemét jelöli. Ezzel ekvivalensen felírható a következő formula, ahol $\otimes$ a tenzorszorzatot jelöli:

$$\frac{\partial E}{\partial W} = w_I \otimes A = w_I A^T.$$

Mivel w_I csak egy helyen vesz fel 1-et és mindenhol máshol 0, a fenti egyenlet A^T mátrix egy sorának értékét adja, melynek dimenziója D. Ebből a következő egyenletet kapjuk W frissítéséhez:

$$v'^{(\text{új})}_{w_I} = v'^{(\text{régi})}_{w_I} - \eta A^T.$$

W többi sorát nem kell változtatnunk ebben a lépésben, mivel a deriváltjuk 0.

Számítási összetettség

Ennek a modellnek a komplexitása $\mathcal{O}(N \cdot D) + \mathcal{O}(2c \cdot D \cdot N)$, mivel a bemenet itt is $1 - N$ kódolással volt reprezentálva, és a kimeneti vektor dimenziója is N, a második rétegben pedig $2c$ szót becsültünk meg [25, 26]. A tanítási fázis hasonlóan zajlik a korábbi CBOW modelléhez, így ebben a formában egy nagyméretű szótár esetén nagyon költséges. A 4.4. bekezdésben láthatunk néhány megoldást arra, hogy hogyan tehető hatékonyabbá ez a modell.

4.3. GloVe modell

A globális vektor modell célja, hogy ötvözze a nyelvfeldolgozáshoz használt eszközök két nagy csoportját, a globális mátrix faktorizációt és a lokális szövegkörnyezetet használó eljárásokat. A modellt a 2014-es publikációja alapján vizsgáljuk meg [30]. Ahelyett, hogy egyszerűen megadnánk az optimalizálni kívánt célfüggvényt, az alábbiakban levezetjük, hogy milyen gondlatmenet alapján lett megalkotva. A faktorizáció során használt mátrixot X-szel fogjuk jelölni, azaz az X_{ij} elem jelentése, hogy a j szó hányszor fordul elő az i szó környezetében. $X_i = \sum_k X_{ik}$-val fejezzük ki, hogy az i szó környezetében hány másik szó fordul elő, míg $P_{ij} = P(j \mid i) = Xij/X_i$ megadja annak a valószínűségét, hogy a j szó megjelenik az i környezetében. A szavak jelentését bizonyos módon kifejezik a definiált együttes előfordulásra vonatkozó valószínűségek, ennek megértéséhez tekintünk egy egyszerű példát. Tegyük fel, hogy a $i =$ jég és a $j =$ gőz szavakra vagyunk kíváncsiak. Mivel mindkét fogalom jelentése hasonlóan szorosan kapcsolódik $k =$ víz szóhoz, a P_{ik} és P_{jk} valószínűségek hasonlóan nagyok lesznek és ebben az esetben $\frac{P_{ik}}{P_{jk}}$ hányadosuk értéke közel 1 lesz. Ellenben ha vesszük a $k =$ bútor szót, akkor P_{ik} és P_{jk} egyformán kicsik lesznek, mivel egyikük se kötődik hozzá jelentésben. A hányadosuk itt is nagyjából egyet ad. Akkor érdekes a helyzet, ha egy olyan fogalmat tekintünk, ami csak a jég vagy csak a gőz szóhoz kapcsolódik. Például ilyen ha k-nak a szilárd szót választjuk. Ilyenkor P_{ik} értéke nagy lesz, de P_{jk} kicsi, tehát a $\frac{P_{ik}}{P_{jk}}$ is nagy értéket ad. Fordított esetben, ha $k =$ légnemű, $\frac{P_{ik}}{P_{jk}}$ nagyságrendekkel kisebb lesz egynél. Összefoglalva, a vizsgált hányados képes kifejezni a két vizsgált szó egymáshoz való viszonyát, ám csak az 1-től jelentősen eltérő értékek adnak releváns információt ezen szavak eltérésére vonatkozóan. Szeretnénk kihasználni ezt a tulajdonságot, ezért bevezetünk egy F függvényt az említett hányados kiszámításához, melyben $w \in \mathbb{R}^N$ szóvektorok és $\widetilde{w} \in \mathbb{R}^N$ különböző szövegkörnyezetbeli szóvektorok, N pedig a szótár méretét jelöli.

$$F\left(w_i, w_j, \widetilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}. \tag{6}$$

Az egyenletben a jobb oldalon szereplő értéket megkaphatjuk a korpuszból, míg a baloldali kifejezést többféleképpen is definiálhatjuk. Szeretnénk ha F a szavak vektorterében reprezentálná az információkat, amiket a hányados tartalmaz. Mivel a vektorterek lineáris struktúrák, és az argumentumok is a kívánt vektortér elemei, tekinthetjük a két vizsgált szóvektorunk különbségét is a következő mó-

don:

$$F\left(w_i - w_j, \widetilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}. \tag{7}$$

Mivel a keresett függvény nagyon összetett is lehet, ha vesszük a 7. egyenletben szereplő argumentumok skaláris szorzatát, elkerülhetjük, hogy a elveszítsük a linearitást a kiértékelés során.

$$F\left((w_i - w_j)^T \widetilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}.$$

Ahhoz, hogy F megfelelő legyen, ki kell elégítenie az együttes előfordulás mátrixban megjelenő szimmetriát. Azaz, egy szónak és egy a környezetében megjelenő szónak felcserélhetőnek kell lennie, hiszen, ha i szerepel j környezetében, akkor j is szerepel az i-jében. Tehát X és X^T mellett szükségünk van arra, hogy w és $\widetilde{w}$ szerepe szimmetrikus legyen. Ennek eléréséhez megköveteljük F homomorfizmusát $(\mathbb{R}, +)$ és $(\mathbb{R}_{>0}, \cdot)$ csoportok között. Így átalakítva az előző képletet megkapjuk, hogy

$$F\left((w_i - w_j)^T \widetilde{w}_k\right) = \frac{F(w_i^T \widetilde{w}_k)}{F(w_j^T \widetilde{w}_k)}, \tag{8}$$

ahonnan adódik, hogy

$$F\left(w_i^T \widetilde{w}_k\right) = P_{ik} = \frac{X_{ik}}{X_i}.$$

A 8. egyenlet megoldása, hogy F az exponenciális függvény, vagy ezzel egyenértékű, hogy $w_i^T \widetilde{w}_k = \log P_{ik} = \log X_{ik} - \log X_i$. Mivel így még nem szimmetrikus a jobb oldal, szeretnénk a $\log X_i$ tagot eltüntetni. Emiatt egy b_i eltolást hozzárendelünk w_i-hez kapcsolódóan, és a szimmetria miatt egy $\widetilde{b}_k$ tagot is $\widetilde{w}_k$-hoz: $w_i^T \widetilde{w}_k + b_i + \widetilde{b}_k = \log X_{ik}$. Annak érdekében, hogy elkerüljük, hogy az egyenletben a logaritmus argumentumában 0 legyen, eltolatjuk 1-gyel: $\log(X_{ik}) \to \log(X_{ik} + 1)$.

A kapott modellnek a hátránya, hogy egyenlően súlyozza az együttes előfordulásait a szavaknak, attól függetlenül, hogy ezek milyen gyakorisággal történnek meg. Ennek eredménye, hogy a ritka adatok zajosak és kevesebb információt tartalmaznak, mint a gyakoriak. Emellett az X mátrix nagyon ritka lesz. Ahhoz, hogy ezeket elkerüljük tekintsük a következő legkisebb négyzetek módszerét használó regressziós modellt:

$$J = \sum_{i,j=1}^{N} f\left(X_{ij}\right)\left(w_i^T \widetilde{w}_k + b_i + \widetilde{b}_k - \log X_{ik}\right)^2, \tag{9}$$

ahol f egy olyan súlyfüggvény, amelyre a következő feltételek teljesülnek:

- $f(0) = 0$. Ha f folytonos függvény, akkor x elég gyorsan tart 0-hoz,

hogy $\lim_{x \to 0} f(x) \log^2 x$ véges legyen.

- $f(x)$-nek nemcsökkenőnek kell lennie, ahhoz hogy elkerüljük a ritka együttes előfordulások túlsúlyozását.
- $f(x)$-nek kicsinek kell lennie a nagy x értékekre, hogy a gyakori együttes előfordulások ne legyenek túlsúlyozva.

A gyakorlatban a következő módon megadott függvények jól működtek a vizsgált feladatokon:

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{ha } x < x_{\max} \\ 1 & \text{egyébként.} \end{cases}$$

A modell alkotói tapasztalati eredmények alapján az α értékét 3/4-nek, $x_{\max}$ értékét 100-nak választották.

Számítási összetettség

A modell (9. egyenlet) komplexitásának vizsgálatakor érdemes először az $f(x)$ súlyfüggvényt tekinteni, ugyanis, ha az i és j szóra az együttes előfordulási mátrix eleme nulla, akkor a súlyuk is nulla lesz. Emiatt a modell számítási összetettségét az X nemnulla elemeinek száma befolyásolja. Mivel X egy $N \times N$ dimenziójú mátrix, a komplexitásunk nem lehet nagyobb, mint $\mathcal{O}(N^2)$. Azonban ez a becslés nem kielégítő, mivel a szótár mérete hatalmas lehet, akár több milliárd szót is tartalmazhat. Szeretnénk tehát egy jobb közelítést találni X nemnulla elemeinek a számára. Ehhez használni fogjuk a gyakorisági rangot, ami egyszerűen annyit jelent, hogy a gyakoriságuk szerint rangsoroljuk a szavakat, azaz a leggyakoribb szó gyakorisági rangja 1 lesz, a második leggyakoribb szóé 2, és így tovább.

Tegyük fel, hogy az i és j szavak együttes előfordulása modellezhető az r_{ij} gyakorisági rangjuk segítségével a következő módon:

$$X(ij) = \frac{k}{(r_{ij})^{\alpha}}. \tag{10}$$

A lokális szövegkörnyezetet használó modellek komplexitása általában a korpusz méretével arányos - jelölje ezt $|C|$ -, ezért összehasonlítjuk, hogy a GloVe modell, hogyan viszonyul hozzá. Ha tekintjük az együttes előfordulási mátrix elemeinek az összegét, akkor láthatjuk, hogy ez arányos a korpusz méretével, hiszen minden szó előfordulását számításba vesszük a környezetében szereplő

szavakkal. Használva az előbbi feltevésünket a következő egyenletet kapjuk:

$$|C| \sim \sum_{ij} X(ij) = \sum_{r=1}^{|X|} \frac{k}{r^{\alpha}},$$

ahol $|X|$ a legnagyobb rangot jelöli, ami pontosan X elemeinek a számát adja meg. Mivel az $X(ij) \neq 1$ elemeket vizsgáljuk, r értéke maximum $k^{1/\alpha}$ lehet, vagyis $|X| = k^{1/\alpha}$ teljesül. A fenti egyenletben egy általánosított harmonikus szám k-szorosát kaptuk, ezért legyen

$$H_{|X|,\alpha} = \sum_{r=1}^{|X|} \frac{1}{r^{\alpha}}.$$

A [2] irodalom alapján kibővítjük a harmonikus számot, hogy feltárjuk $|X|$ és $|C|$ viszonyát. Kiemeljük, hogy a bővítést azért alkalmazhatjuk, mivel minket csak nagy méretű korpusz/szótár esetén érdekel a kapcsolat, hiszen a modell komplexitására vagyunk kíváncsiak. A bővített szám a következőképpen írható fel:

$$H_{x,s} = \frac{x^{1-s}}{1-s} + \zeta(s) + \mathcal{O}(x^{-s}), \qquad \text{ha } s > 0,\ s \neq 1,$$

ahol $\zeta(s) = \sum_{n=1}^{\infty} 1/n^s$ a Riemann-féle zéta-függvényt jelöli. Felhasználva ezt és a 10. egyenletet:

$$|C| \sim \frac{|X|}{1-\alpha} + |X|^{\alpha}\zeta(\alpha) + \mathcal{O}(1).$$

Ha $\alpha < 1$, akkor a jobb oldal első tagja határozza meg a GloVe modell komplexitását, viszont, ha $\alpha > 1$, akkor a második tag. Összefoglalva:

$$|X| = \begin{cases} \mathcal{O}(|C|) & \text{ha } \alpha < 1 \\ \mathcal{O}(|C|^{1/\alpha}) & \text{ha } \alpha > 1. \end{cases}$$

A [30] cikk alapján α-t 1,25 - nek választva jól modellezhető az együttes előfordulási mátrix, ami a számítási összetettségre vonatkozóan azt jelenti, hogy a GloVe modell messze hatékonyabb, mint az $\mathcal{O}(N^2)$ alapú modellek és valamivel felülmúlja a lokális környezetet használó modelleket is a $\mathcal{O}(|C|^{0,8})$-as komplexitásával.

Kapcsolat a Skip-gram modellel

A modell bevezetésekor említettük, hogy a modell célja, hogy a globális és a lokális információkat használó eljárásokat egybevonja, ezzel javítva őket. A 4.2. fejezetben láthattuk, hogy a Skip-gram modell is egy lokális szövegkörnyezetet használó eljárás. Emellett a korpusz statisztikai adatait használja, hogy meghatározza a szavak vektor reprezentációját és a tanítás felügyelet nélkül zajlik. Figyelembe véve ezeket a hasonlóságokat a GloVe modellel, megvizsgáljuk a két modell kapcsolatát.

Jelölje Q_{ij} azt a valószínűséget, hogy az i szó környezetében feltűnik a j. Ez a Skip-gram modell esetében a következőt adja az 5. egyenlet alapján:

$$Q_{ij} = \frac{\exp(w_i^T \widetilde{w}_j)}{\sum_{k=1}^{N} \exp(w_i^T \widetilde{w}_j)}. \tag{11}$$

A Skip-gram célfüggvényét (4. egyenlet) átalakítva minimalizálási feladatra a következő modellt kapjuk:

$$J = -\sum_{i=1}^{N} \sum_{\substack{|j|\leq c \\ j\neq 0}} \log Q_{ij}.$$

A szoftmax függvény normalizációs tényezőjét nagyon költséges kiszámolni az összegzés minden kifejezésére, ezért inkább Q_{ij} egy közelítését fogjuk használni. Alakítsuk át az előző egyenletet az együttes megjelenést reprezentáló X mátrixot használva:

$$J = -\sum_{i=1}^{N} \sum_{j=1}^{N} X_{ij} \log Q_{ij}.$$

Ha felhasználjuk a $P_{ij} = X_{ij}/X_i$-t, a következőt kapjuk:

$$J = -\sum_{i=1}^{N} X_i \sum_{j=1}^{N} P_{ij} \log Q_{ij} = \sum_{i=1}^{N} X_i H(P_i, Q_i),$$

ahol H a kereszt-entrópiát jelöli. Az eredményt nevezhetjük "globális skip-gram" modellnek, melynek az előnye, hogy közvetlenül optimalizálható (hasonlít a 9. egyenletre). Sajnos emellett hátrányai is akadnak, például, hogy a kereszt-entrópia egy olyan távolsági mérték a valószínűségi eloszlások között, amely a hosszú farokkal rendelkező eloszlásokat rosszul modellezi és túl nagy súllyal veszi figyelembe a csekély valószínűségű eseményeket.Emellett figyelni kell Q normalizálására is, hogy a mérték korlátos maradjon ami jelentősen megnöveli

a számítási összetettséget, hiszen az egész szótár feletti összegzésre van szükség (lásd a 11. egyenletet). Figyelembe véve eme hátrányokat érdemes lehet egy másik mérték használata, például a legkisebb négyzetek jó választásnak tűnik, mivel kihagyja a normalizálást:

$$\widetilde{J} = \sum_{i,j} X_i (\widetilde{P}_{ij} - \widetilde{Q}_{ij})^2,$$

ahol $\widetilde{P}_{ij} = X_{ij}$ és $\widetilde{Q}_{ij} = \exp(w_i^T \widetilde{w}_j)$. Mivel itt X_{ij} túl nagy értékeket vehet fel ezzel megnehezítve az optimalizálást, tekintsük a $\widetilde{P}$ és $\widetilde{Q}$ logaritmusainak a négyzetes hibáját helyette:

$$\widetilde{J} = \sum_{i,j} X_i (\log \widetilde{P}_{ij} - \log \widetilde{Q}_{ij})^2 = \sum_{i,j} X_i (w_i^T \widetilde{w}_j - \log X_{ij})^2.$$

Fontos megjegyezni, hogy az X_i-vel való súlyozás nem feltétlenül ad optimális eredményt. Sőt, ha csökkentjük a súlyozás értékét a gyakori szavak esetén az javít az eredményen. Általánosabban véve használható a következő súly függvény, amit a 9. egyenletben definiáltunk:

$$J = \sum_{i,j} f\left(X_{ij}\right) \left(w_i^T \widetilde{w}_j - \log X_{ik}\right)^2.$$

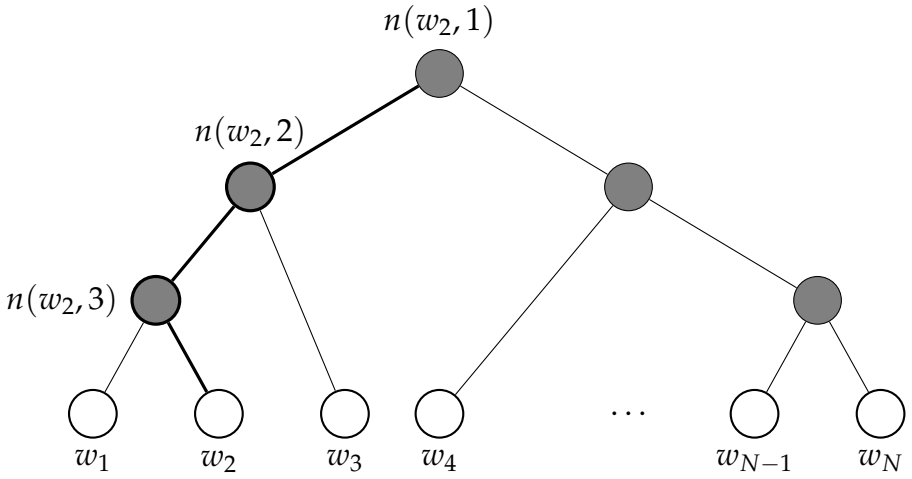
4.4. Nyelvi modellek optimalizálása

Korábban megvizsgáltuk a CBOW és a Skip-gram modelleket. Láthattuk, hogy a számítási összetettségüket jelentősen növeli, hogy minden fázisban minden kimeneti vektort frissítünk (ezt jelöltük korábban v'_{w_j}-vel). Az alapötlet amivel csökkenthető a komplexitás, hogy csak a legbefolyásolóbb kimenetei vektorokat újítjuk meg a fázisokban. A folytatásban leírt két eljárásnak a célja, hogy megvalósítást nyújtson erre a problémára.

4.4.1. Hierarchikus szoftmax

A hierarchikus szoftmax modell egy megoldás arra, hogy optimalizáljuk a kimeneti vektorok (v'_j) javításának a számítási igényét a Skip-gram és a Continuous Bag-of-Word modellekben [26, 32, 10]. Ebben az esetben modell bináris fában kódolja az összes szótárbeli szót. Ez az N szó a levelekben helyezkedik el, ezeken kívül még $N-1$ belső csúcs van (a gyökeret is beleszámítva).

A belső csúcsokra a következő jelölést használjuk: $n(w, j)$, ahol a j-edik csúcsot jelöli a gyökérből a w szóhoz vezető úton. Megjegyezzük, hogy mindig a gyökér számít az első csúcsnak és minden csúcsba egyértelmű út vezet a gyökérből. Fel fogjuk használni ezeket az utakat, hogy megbecsüljük a valószínű-



ségét a levelekben szereplő szavaknak. A hierarchikus szoftmax modell egyik eltérése az előzőhöz képest, hogy itt nem lesznek kimeneti vektor reprezentációi a szavaknak, hanem a belső csúcsokhoz definiálunk kimeneti vektorokat: $v'_{n(w,j)}$ az $n(w, j)$ belső csúcshoz tartozó kimeneti vektor. Annak a valószínűségét, hogy a w szó az elvár kimenet, a következőképpen definiáljuk:

$$p(w \mid w_O) = \prod_{j=1}^{L(w)-1} \sigma\left(b(n(w, j+1) = \text{ch}(n(w, j))) \cdot v'^T_{n(w,j)} \cdot h\right),$$

ahol $L(w)$ a gyökértől a w-be vezető út hosszát jelöli, $\text{ch}(n)$ az n csúcs baloldali gyerekét, h a rejtett réteg kimeneti értéke és $b(x)$ értéke 1, ha x igaz, különben -1. σ a szoftmax függvény, azaz $\sigma(x) = 1/1 + \exp(-x)$.

Tekintsünk egy példát, amelyen bemutatjuk, hogy mi indokolja az előbbi feltételes valószínűség meghatározását. Tekintsük az ábrát és tegyük fel, hogy annak a valószínűségét keressük, hogy a w_2 a megjósolandó szó egy adott bemenet mellett. Erre a valószínűségre úgy tekintünk, hogy egy véletlen séta során mekkora a valószínűsége annak, hogy a gyökérből a w_2 levélbe jutunk. Ahhoz, hogy ezt meghatározzuk, szükségünk van arra, hogy milyen valószínűséggel lépünk egy belső csúcsból jobbra vagy balra. Definiáljuk a n belső csúcsból való balra lépé valószínűségét a következőképpen:

$$p(n, \text{left}) = \sigma(v'^T_n \cdot h).$$

Ekkor annak a valószínűsége, hogy jobbra megyünk:

$$p(n, \text{right}) = 1 - \sigma(v'^T_n \cdot h) = \sigma(-v'^T_n \cdot h).$$

Ha az ábrán szereplő utat tekintjük, akkor a keresett valószínűséget a következő

módon kapjuk meg:

$$\begin{aligned} p(w_2 \mid w_O) &= p(n(w_2,1),\text{left}) \cdot p(n(w_2,2),\text{left}) \cdot p(n(w_2,3),\text{right}) \\ &= \sigma(v'^T_{n(w_2,1)} \cdot h) \cdot \sigma(v'^T_{n(w_2,2)} \cdot h) \cdot \sigma(-v'^T_{n(w_2,3)} \cdot h), \end{aligned}$$

ami egybeesik a bevezetett feltételes valószínűséggel.

Ahhoz, hogy meghatározzuk a belső csúcsok reprezentációját tekintsük a legegyszerűbb változatát a modellnek, amikor egy adott szóból szeretnénk következtetni a rákövetkezőre. Először is bevezetünk két jelölést: $v'_j := v'_{n(w,j)}$ és $b(\cdot) = b(n(w, j+1) = \text{ch}(n(w, j))$. A hibafüggvény a következő értéket veszi fel:

$$E = -\log p(w = w_O \mid w_I) = -\sum_{j=1}^{L(w)-1} \log \sigma\left(b(\cdot) \cdot v'^T_j h\right).$$

Vegyük E deriváltját $v'^T_j h$ szerint:

$$\frac{\partial E}{\partial v'^T_j h} = \left(\sigma\left(b(\cdot) \cdot v'^T_j h\right) - 1\right) b(\cdot) = \sigma\left(b(\cdot) \cdot v'^T_j h\right) - t_j,$$

ahol $t_j = 1$, ha $b(\cdot) = 1$, különben -1.

Vegyük e deriváltját a belső csúcs vektor reprezentációjára vonatkozóan:

$$\frac{\partial E}{\partial v'^T_j} = \frac{\partial E}{\partial v'^T_j h} \cdot \frac{\partial v'^T_j h}{\partial v'^T_j} = (\sigma\left(b(\cdot) \cdot v'^T_j h\right) - t_j) \cdot h.$$

Ebből a következő egyenlet adódik a súlyok javítására:

$$v'^{(\text{új})}_j = v'^{(\text{régi})}_j - \eta(\sigma\left(b(\cdot) \cdot v'^T_j h\right) - t_j) \cdot h \qquad \forall j = 1, 2, \ldots, L(w) - 1.$$

A bemeneti réteg és a rejtett réteg közötti súlyok frissítéséhez ki kell számolnunk a következőt:

$$\begin{aligned} \frac{\partial E}{\partial h} &= \sum_{j=1}^{L(w)-1} \frac{\partial E}{\partial v'^T_j h} \cdot \frac{\partial v'^T_j h}{\partial h} = \sum_{j=1}^{L(w)-1} (\sigma\left(b(\cdot) \cdot v'^T_j h\right) - t_j) \cdot v'_j, \\ A &:= \sum_{j=1}^{L(w)-1} (\sigma\left(b(\cdot) \cdot v'^T_j h\right) - t_j) \cdot v'_j. \end{aligned}$$

Ahonnan a korábbi levezetést használva megkaphatjuk az átsúlyozáshoz szükséges egyenleteket.

4.4.2. Negatív mintavétel

Egy másik lehetőség Skip-gram és a CBOW modellek fejlesztésére a negatív mintavétel használata, aminek a lényege, hogy nem az összes kimeneti vektort javítjuk, hanem csak egy részét. Azaz egy mintavételezést hajtunk végre rajtuk, és csak a kiválasztott példányokat fogjuk frissíteni [26, 32, 10].

Az egyszerűség kedvéért most is tekintsük azt az esetet, amikor egyetlen bemeneti w_I szóból egyetlen szót szeretnénk megjósolni, ahol a w_O szót kapjuk eredményül. Ha jól működik a modellünk, akkor igaz lesz az, hogy w_I és w_O is a vizsgált korpuszból ered, hiszen a CBOW és a Skip-gram modell esetén is a vizsgált szöveg szavából következtetünk egy másik kifejezésére. Ezt felhasználva a negatív mintavételezésnél azt szeretnénk maximalizálni, hogy a $w_I - w_O$ párok megtalálhatóak a korpuszban. Formálisan $p(D = 1 \mid w_I, w_O)$ jelöli azt, hogy a D szövegkörnyezet tartalmazza w_I és w_O szavakat, még $p(D = 0 \mid w_I, w_O) = 1 - p(D = 1 \mid w_I, w_O)$ jelentése, hogy nem D-ből származik a bemeneti - kimeneti szópár. Ebből kapjuk a következő célfüggvényt:

$$\begin{aligned}\max \prod_{(w_I,w_O)\in D} p(D = 1 \mid w_I, w_O) &= \max \log \prod_{(w_I,w_O)\in D} p(D = 1 \mid w_I, w_O) \\ &= \max \sum_{(w_I,w_O)\in D} \log p(D = 1 \mid w_I, w_O).\end{aligned}$$

Használva a szoftmax függvényt és a szavak v' kimeneti vektor reprezentációját, a következő célfüggvényt kapjuk meg:

$$\sum_{(w_I,w_O)\in D} \log \frac{1}{1 + \exp(-v'_{w_I} v'_{w_O})}.$$

Ennek a függvénynek egy triviális megoldása, ha $p(D = 0 \mid w_I, w_O) = 1$ minden szópárra azáltal, hogy $v'_{w_I} = v'_{w_O}$ és $v'_{w_I} \cdot v'_{w_O} = K$, ahol K elég nagy szám. Viszont szeretnénk elkerülni ezt az esetet, ezért módosításra van szükségünk. Ha használunk a célfüggvényben olyan szópárokat is, amiknél $p(D = 0 \mid w_I, w_O)$ alacsony, kiküszöbölhetjük ezt a problémát. Ehhez egyszerűen elég, ha olyan szópárokat is számításba veszünk, amik nem a D korpuszból valóak. Erre egy módszer, ha véletlenszerűen generálunk egy szópárokból álló D' halmazt, melynél feltételezzük, hogy a szópárok nem részei D-nek. Az elnevezés is innen ered,

hiszen negatív párosokból is válogatunk a mintavételezés során:

$$\max \prod_{(w_I,w_O)\in D} p(D=1 \mid w_I, w_O) \prod_{(w_I,w_O)\in D'} p(D=0 \mid w_I, w_O) =$$
$$\max \prod_{(w_I,w_O)\in D} p(D=1 \mid w_I, w_O) \prod_{(w_I,w_O)\in D'} (1 - p(D=1 \mid w_I, w_O)) =$$
$$\max \sum_{(w_I,w_O)\in D} \log p(D=1 \mid w_I, w_O) + \sum_{(w_I,w_O)\in D'} \log(1 - p(D=1 \mid w_I, w_O)) =$$
$$\max \sum_{(w_I,w_O)\in D} \log \frac{1}{1+\exp(-v'_{w_I} v'_{w_O})} + \sum_{(w_I,w_O)\in D'} \log \left(1 - \frac{1}{1+\exp(-v'_{w_I} v'_{w_O})}\right)$$
$$\max \sum_{(w_I,w_O)\in D} \log \frac{1}{1+\exp(-v'_{w_I} v'_{w_O})} + \sum_{(w_I,w_O)\in D'} \log \frac{1}{1+\exp(-v'_{w_I} v'_{w_O})},$$

melyen alkalmazva a $\sigma(x) = \frac{1}{1+\exp(-x)}$ jelölést végül megkapjuk, hogy:

$$\max \sum_{(w_I,w_O)\in D} \log \frac{1}{1+\exp(-v'_{w_I} v'_{w_O})} + \sum_{(w_I,w_O)\in D'} \log \frac{1}{1+\exp(-v'_{w_I} v'_{w_O})} =$$
$$\max \sum_{(w_I,w_O)\in D} \log \sigma(v'_{w_I} \cdot v'_{w_O}) + \sum_{(w_I,w_O)\in D'} \log \sigma(v'_{w_I} \cdot v'_{w_O})$$

5. Egynyelvű analóg kérdések

Most tekintsünk egy példa feladatot arra, ami a vektortér modellek egy érdekes tulajdonságát vizsgálja. Korábban is említettük, hogy a tárgyalt vektortér modelleknek meg van az a tulajdonsága, hogy a szavak közti távolsággal jellemezni lehet a szavak hasonlóságát. Ennek a mérésére olyan speciális szókészleteket szoktak használni, amelyek szópárok távolságait hasonlítja össze. Most megadunk pár példát a gyakran vizsgált szópár típusokra:

- főváros - ország: Budapest - Hungary
- ország - pénznem: Hungary - forint
- férfi - női megfelelők: king - queen
- melléknév fokozás: bad - worse
- egyes szám - többes szám: banana - bananas

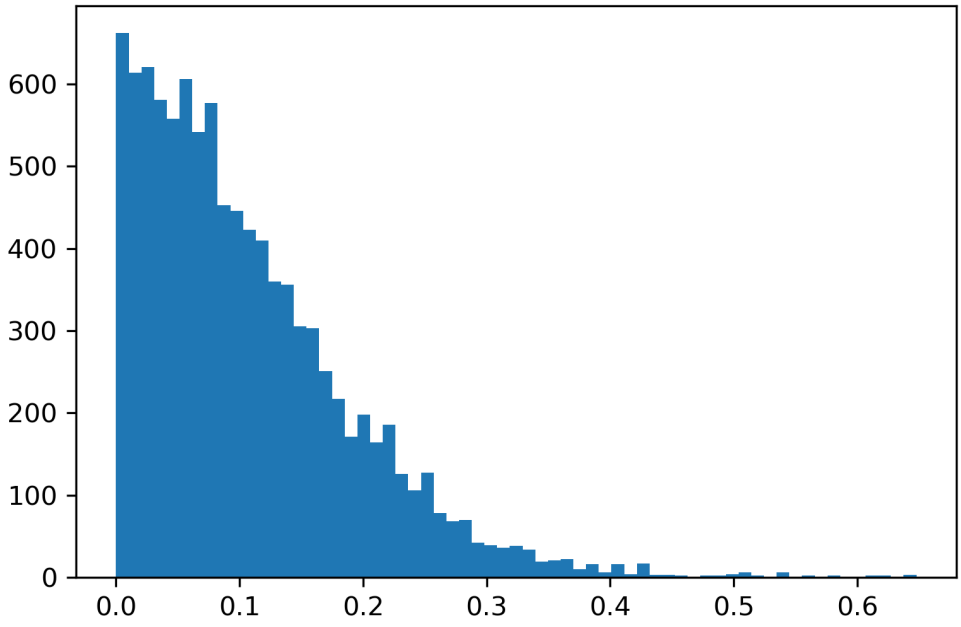
Az elemzés során több hasonló kategóriába eső szópárt hasonlítanak össze egymással, hogy a köztük külön-külön megjelenő távolságok közel azonosak-e. Például azt várnánk, hogy $v(\text{king}) - v(\text{queen})$ értéke megegyezik a $v(\text{policeman}) - v(\text{policewoman})$-vel, ahol $v(\text{x})$ az x szó egy vektor reprezentációját jelöli.

A dolgozatomban én azt vizsgáltam, hogy ha ötvöznénk egy lokális szövegkörnyezetet használó vektort tér modellt – a továbbiakban word2vec – egy globális információkat használóval – glove –, akkor jobb eredményt érhetnék-e el az analóg kérdések vizsgálata során. A folyamat során előre tanított szóvektorokat használtam:

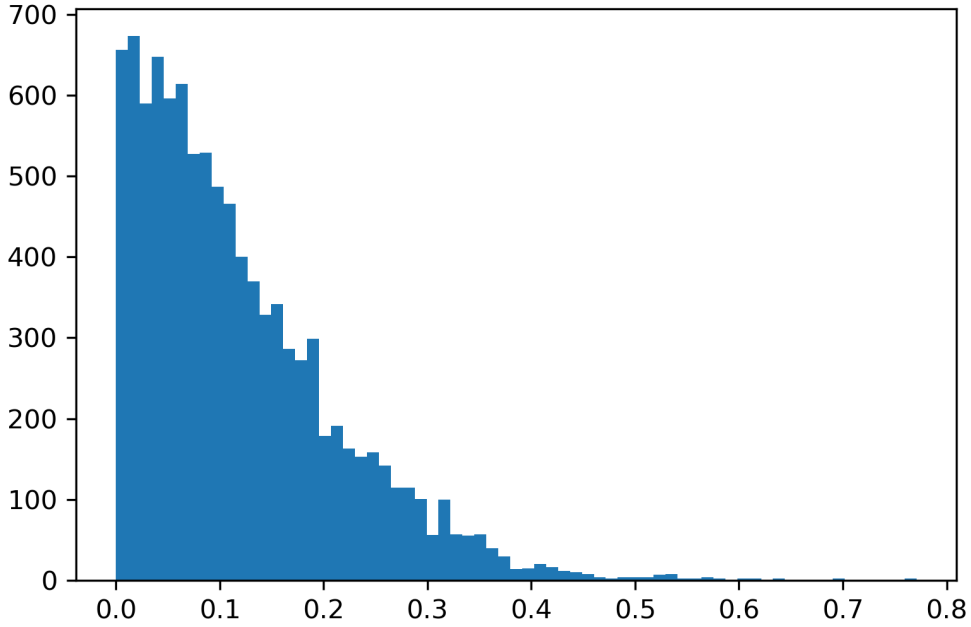
- word2vec: 100 milliárd szavas adathalmazon tanított 300 dimenziós vektorok: https://code.google.com/archive/p/word2vec/
- glove: 400 ezer szóból tanult 100 dimenziós vektorok https://nlp.stanford.edu/projects/glove/

A word2vec modellt is korlátoztam egy csak 400 ezer szót használó verzióra. Ennek a fő oka az volt, hogy reálisabb képet kapjunk a glove és a word2vec modellek összehasonlítása során, a másik pedig, hogy csökkentsük a számítási kapacitást. Utóbbira azért volt szükség, mivel az elemzéseket a saját gépemen végeztem:

- Processzor: Intel Core i3, 2.4Ghz
- Operációs rendszer: Microsoft Windows 10, 64 bites



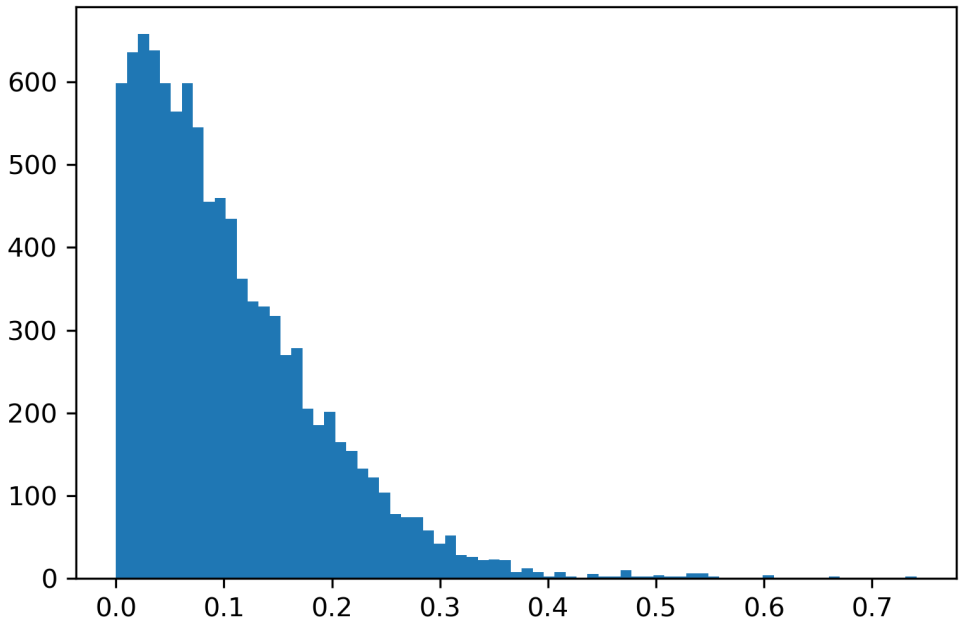
10. ábra. word2vec modellre kapott eredmények, melynek x tengelye a szópárok között mért eltérést adja meg, függőleges tengelye pedig azt fejezi ki, hogy egy rögzített eltérés mellett hány darab vizsgált szópár-szópár kapcsolatra kaptuk ezt az eredményt.



11. ábra. glove modellre kapott eredmények, melynek x tengelye a szópárok között mért eltérést adja meg, függőleges tengelye pedig azt fejezi ki, hogy egy rögzített eltérés mellett hány darab vizsgált szópár-szópár kapcsolatra kaptuk ezt az eredményt.

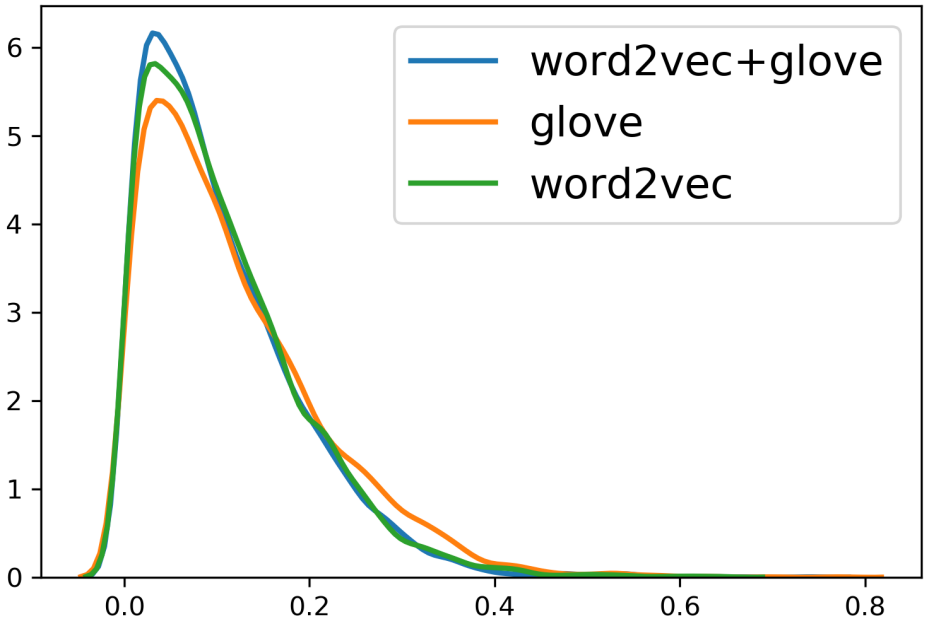
- Memória: 4 Gb memória

A 10., 11., 12. ábrákon olyan hisztogramokat láthatunk, melyek vízszintes tengelye azt fejezi ki, hogy mekkora volt a különbség a szópár - szópár távolságok között. A feldolgozás során először vettem a szóvektor párok koszinusz hasonlóságát, tehát például cos_ sim(v(queen),v(king)) és cos_ sim(v(girl), v(boy)), majd kiszámítottam az így kapott értékek közti eltérést. A glove + word2vec modell esetében összefűztem a két modellből nyer szóvektorokat és ennek az eredményét használtam egy új szóvektor reprezentációként.



12. ábra. word2vec + glove modellre kapott eredmények, melynek x tengelye a szópárok között mért eltérést adja meg, függőleges tengelye pedig azt fejezi ki, hogy egy rögzített eltérés mellett hány darab vizsgált szópár-szópár kapcsolatra kaptuk ezt az eredményt.

A 13. ábrán az összesített eredményeket láthatjuk, melyből leolvasható, hogy a hibrid modell jobb eredményeket tudott elérni az analóg vizsgálat során. Abból vonhatjuk le ezt a következtetést, hogy az összetett modell esetén sokkal több kisebb eltérést adó eredményünk volt, ami abból látható, hogy a nulla értékhez közel kimagaslik a görbe a többi modellhez viszonyítva. Úgy gondolom, hogy a későbbiekben érdemes lehet többféle paraméter beállítással megvizsgálni további modelleket is.



13. ábra. Összesítve a három hisztogramra külön-külön illesztett görbe, melynek vízszintes tengelye a szópárok közötti eltérést méri, a függőleges pedig, hogy mennyi vizsgált kérdésre kaptuk az adott eltérést eredményül. A függőleges tengely százas nagyságrendben értendő.

Hivatkozások

[1] Ossama Abdel-Hamid, Li Deng, and Dong Yu. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *Interspeech 2013*. ISCA, August 2013.

[2] T.M. Apostol. *Introduction to Analytic Number Theory*. Undergraduate Texts in Mathematics. Springer New York, 2013.

[3] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1044–1054, 2013.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994.

[6] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. *CoRR*, abs/1603.07252, 2016.

[7] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398, 2011.

[8] Li Deng and Dong Yu. Deep learning: Methods and applications. Technical report, May 2014.

[9] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over freebase with multi-column convolutional neural networks. In *ACL*, 2015.

[10] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1319–III–1327. JMLR.org, 2013.

[13] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[14] Alex Graves, Santiago Fernández, and Faustino Gomez. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *In Proceedings of the International Conference on Machine Learning, ICML 2006*, pages 369–376, 2006.

[15] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013.

[16] Simon S. Haykin. *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition, 2009.

[17] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc., 2015.

[18] Gauri Jain, Manisha, and Basant Agarwal. An overview of rnn and cnn techniques for spam detection in social media. *IJARCSSE*, 6(10):126–132, 2016.

[19] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *CoRR*, abs/1412.1058, 2014.

[20] Rie Johnson and Tong Zhang. Semi-supervised convolutional neural networks for text categorization via region embedding. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 919–927. Curran Associates, Inc., 2015.

[21] Szatmári József. *Modellek a geoinformatikában*. Szegedi Tudományegyetem; Debreceni Egyetem; Pécsi Tudományegyetem, 2013. https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011_0025_geo_4/adatok.html.

[22] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[23] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. *CoRR*, abs/1701.02810, 2017.

[24] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. *CoRR*, abs/1708.00107, 2017.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[27] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[28] Ramesh Nallapati, Bing Xiang, and Bowen Zhou. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023, 2016.

[29] K. O'Shea and R. Nash. An Introduction to Convolutional Neural Networks. *ArXiv e-prints*, November 2015.

[30] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[31] Adarsh Rajesh and Megha Mantur. Eyeball gesture controlled automatic wheelchair using deep learning. *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, pages 387–391, 2017.

[32] Xin Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014.

[33] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685, 2015.

[34] Tara N. Sainath, Brian Kingsbury, Abdel-rahman Mohamed, George E. Dahl, George Saon, Hagen Soltau, Tomas Beran, Aleksandr Y. Aravkin, and Bhuvana Ramabhadran. Improvements to deep convolutional neural networks for lvcsr. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 315–320, 2013.

[35] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997.

[36] Xudong Sun, Pengcheng Wu, and Steven C.H. Hoi. Face detection using deep learning: An improved faster rcnn approach. *Neurocomputing*, 299:42 – 50, 2018.

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[38] V. N. T. Truong, C. K. Yang, and Q. V. Tran. A translator for american sign language to text and speech. In *2016 IEEE 5th Global Conference on Consumer Electronics*, pages 1–2, Oct 2016.

[39] Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence - video to text. *CoRR*, abs/1505.00487, 2015.

[40] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

[41] Ye Zhang Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015.

[42] Tingmin Wu, Shigang Liu, Jun Zhang, and Yang Xiang. Twitter spam detection based on deep learning. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '17, pages 3:1–3:8, New York, NY, USA, 2017. ACM.

[43] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632, 2014.

[44] Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *CoRR*, abs/1707.07435, 2017.

[45] Ying Zhang, Mohammad Pezeshki, Philemon Brakel, Saizheng Zhang, César Laurent, Yoshua Bengio, and Aaron C. Courville. Towards end-to-end speech recognition with deep convolutional neural networks. *CoRR*, abs/1701.02720, 2017.

NYILATKOZAT

Név: Szögi Viktória Rózsa
ELTE Természettudományi Kar, Szak: Alkalmazott matematikus mesterképzés
NEPTUN azonosító: BHQ3AO
Szakdolgozat címe: Szövegfeldolgozás gépi tanulási módszerekkel

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló munkám eredménye, saját szellemi termékem, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2018

a hallgató aláírása