# Teendők listája


Eötvös Loránd Tudományegyetem

Informatikai Kar

Információs Rendszerek

Tanszék

# Szemantikus reprezentáció magyar nyelv esetén

*Témavezető:*
Grad-Gyenge László
egyetemi tanársegéd

*Szerző:*
Kántor Attila
programtervező informatikus MSc

*Budapest, 2020*

Az eredeti szakdolgozati / diplomamunka témabejelentő helye.

# Tartalomjegyzék

**1. Bevezetés** **3**

**2. Előzmények** **4**

2.1. Reprezentáció a szavak szintjén . . . . . 4

2.1.1. Szótár keresés . . . . . 5

2.1.2. Valószínűség alapú ábrázolás . . . . . 5

2.1.3. Szóvektorok . . . . . 6

2.2. Reprezentáció a mondatok és magasabb nyelvi elemek szintjén . . . . 12

2.2.1. Mondatvektorok . . . . . 12

2.2.2. Dokumentumszintű reprezentáció . . . . . 16

2.3. Transfer learning . . . . . 17

**3. Felhasználói dokumentáció** **19**

3.1. Felsorolások . . . . . 19

3.1.1. Szoros térközű felsorolások . . . . . 20

3.2. Képek, ábrák . . . . . 21

3.2.1. Képek szegélyezése . . . . . 21

3.2.2. Képek csoportosítása . . . . . 22

3.3. Táblázatok . . . . . 23

3.3.1. Sorok és oszlopok egyesítése . . . . . 23

3.3.2. Több oldalra átnyúló táblázatok . . . . . 23

**4. Fejlesztői dokumentáció** **26**

4.1. Tételek, definíciók, megjegyzések . . . . . 26

4.1.1. Egyenletek, matematika . . . . . 27

4.2. Forráskódok . . . . . 28

4.2.1. Algoritmusok . . . . . 29

**5. Összegzés** **30**

**A. Szimulációs eredmények** **31**

**Irodalomjegyzék** **33**

**Ábrajegyzék** **33**

**Táblázatjegyzék** **34**

**Forráskódjegyzék** **35**

# 1. fejezet

# Bevezetés

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In eu egestas mauris. Quisque nisl elit, varius in erat eu, dictum commodo lorem. Sed commodo libero et sem laoreet consectetur. Fusce ligula arcu, vestibulum et sodales vel, venenatis at velit [**dahl1972structured**]. Aliquam erat volutpat. Proin condimentum accumsan velit id hendrerit. Cras egestas arcu quis felis placerat, ut sodales velit malesuada. Maecenas et turpis eu turpis placerat euismod.[1]

Aliquam suscipit dignissim tempor. Praesent tortor libero, feugiat et tellus porttitor, malesuada eleifend felis. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus [**cormen2009algorithms**, **krasner1988mvc**]. Nullam eleifend imperdiet lorem, sit amet imperdiet metus pellentesque vitae. Donec nec ligula urna. Aliquam bibendum tempor diam, sed lacinia eros dapibus id. Donec sed vehicula turpis. Aliquam hendrerit sed nulla vitae convallis. Etiam libero quam, pharetra ac est nec, sodales placerat augue. **dijkstra1979goto** praesent eu consequat purus [**dijkstra1979goto**].

[1] Maecenas a urna viverra, scelerisque nibh ut, malesuada ex.

# 2. fejezet

# Előzmények

Ahogy a nyelvet is szétválaszthatjuk elemeire – például lexéma (szó) , szintagma (szószerkezet) , mondat – , úgy a nyelvi elemeket reprezentáló módszereket is csoportosíthatjuk. Ugyan a nyelvi elemek és a közöttük található nyelvtani kapcsolatok matematikai ábrázolására való törekvés már az előző évszázad közepén megjelent, valódi eredményt csak az elmúlt egy-két évtized tud felmutatni. Az idő során a különböző nyelvi elemek reprezentációs módszerei párhuzamos módon fejlődtek, de a figyelem napjainkban leginkább a magasabb szintekre összpontosul. A mondatokat és a magasabb nyelvi szinteket ábrázoló algoritmusok jobbnál jobb pontosságot mutatnak a különböző NLP feladatok megoldását illetően.

## 2.1. Reprezentáció a szavak szintjén

A szószintű reprezentációs módszerek azt a célt szolgálják, hogy a természetes nyelven írott szöveg szavait numerikusan feldolgozhatóvá tegyék. Ha egy algoritmus képes abszolválni ezt a célt, a számítógép többé már nem karakterláncokat, hanem értelmet is talál a bemenet mögött.

Bár az a gondolat, hogy szavakat matematikailag ábrázoljunk már a '80-as években felütötte a fejét, ezek a módszerek többnyire ritka reprezentációkat eredményeztek. A ritka reprezentációk csak kevés esetben hoznak hatékony megoldást. Számításigényük nagy lehet és néhány feladat esetén a kellő pontosság elérésére is alkalmatlanok.

### 2.1.1. Szótár keresés

A legegyszerűbb technika a szótár keresés, mely során L nyelv minden eleméhez injektív módon egy természetes számot rendelünk. L elemeit szótövezhetjük (*lemmatization*) is, így kisebb szótárat kapunk.

Ez egy kezdetleges és relatíve kis memóriaigényű algoritmus, azonban a neurális hálónkat könnyedén félrevezetheti. A természetes nyelven írott szöveg szavai között csak ritkán találhatunk rendezést. A szótár keresést alkalmazva neurális modellünk fontosabbnak ítélheti azon szavakat, melyek nagyobb azonosítóval rendelkeznek, így ebben az esetben a módszer használhatatlanná válik.

### 2.1.2. Valószínűség alapú ábrázolás

Valószínűség alapú ábrázolásnak nevezünk minden olyan módszert, amely a matematikai valószínűségszámítás eszközeit használja, többnyire eloszlást és gyakoriságot. Ezen reprezentációkat gyenge szemantikai erejük ellenére a mai napig alkalmazzák. Egyszerűek, de memóriaigényük nagy és a tanításuk is körülményes.

#### Gyakoriság és feltételes valószínűség

A csoportot képviselő alapvető algoritmus a gyakoriság alapú leképezés, amely azt az információt veszi figyelembe, hogy a dokumentumok halmazában hányszor szerepel egy adott szó. Használhatunk relatív gyakoriságot is, ha a gyakoriságot elosztjuk a dokumentumok összes szavának számával. Az így kinyert adat akár egyszerűbb szociális média analízisre is alkalmas lehet.

Szekvenciális adatok feldolgozására megfelelő választás lehet a feltételes valószínűség alapú leképezés, mely segítségével képesek lehetünk a következő szó prediktálására az előzőek függvényében.

#### Tf-Idf

A tf-idf egy statisztikai módszer, amely arra hivatott, hogy egy szó előfordulásának fontosságát ragadja meg egy dokumentumban, a dokumentumhalmazban. A modell a Bag Of Words (BOW) modellen alapszik, mely lényege, hogy L szótár esetén egy adott $d \in D$ dokumentumot egy $v \in \{0,1\}^{|L|}$ vektor reprezentál. Ahányszor

előfordul $w \in L$ szó $d$ dokumentumban, $v_{index(w)}$ értéke eggyel növekszik, egyébként marad 0.

A tf-idf két részből áll: *term frequency* és *inverse document frequency*. A végeredmény a két metrika szorzata. Mindkét metrikára több variáció is van, a legnépszerűbb a következő:

**1. Definíció.**

$tf\,(t,d) = \log\left(1 + freq\,(t,d)\right)$, *ahol freq(t,d) t szó gyakorisága d dokumentumban.*

$idf\,(t,D) = \log\left(\dfrac{N}{count\,(d \in D : t \in d)}\right)$, *ahol D dokumentumhalmaz elemszáma N.*

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D)$$

Bár a módszer számításigénye kicsi, továbbá jó választás lehet olyan esetben, ahol dokumentumok hasonlóságát szeretnénk mérni, csak lexikális adatok reprezentálására képes.

**Megjegyzés.** *Természetesen a később bemutatott módszerekben is fellelhetők matematikai valószínűségszámítási eszközök.*

### 2.1.3. Szóvektorok

A valószínűségi modellek jól generalizálnak ritka bemenet esetén, azonban ha sűrűbb a bemenet, azok az algoritmusok bizonyulnak hasznosabbnak, amelyek a szavak jelentéstartalmát is képesek ábrázolni.

Azon feladatok esetén, amikor a szemantikának nagyobb szerepe van – ilyen lehet az írott szöveg érzelmi tartalmának vizsgálata – , nem használhatjuk a fenti technikákat. Olyan reprezentációs módszert kell találnunk, amely képes komplexebb problémákat is megoldani. Ilyen probléma például, ha egy szó több jelentéssel is bír (pl.: mész), a szinonímák és a kontextusfüggő szóhasználat (pl.: víz - $H_2O$).

A szóvektorok részben megoldást nyújthatnak ezen komplikációkra. Szóvektorokat úgy kapunk, ha a lexémákat leképezzük valamely vektortérbe. Ha két szó szemantikai tartalma hasonló, szóvektoruk Euklideszi távolsága kicsi.

### One-hot kódolás

**2. Definíció.** *Legyen $L$ egy $n \in \mathbb{N}$ elemű nyelv. Ekkor $w \in L$ szó one-hot kódolásán $v \in \{0,1\}^n$ vektort értjük, ahol*

$$v_i = \begin{cases} 1, & \text{ha } L_i = w \\ 0, & \text{egyébként.} \end{cases}$$

A *one-hot* kódolás egy egyszerű és nem hatékony reprezentációs módszer, azonban mégis a szóvektorokhoz sorolhatjuk. Minden szóvektort a vektortér egy-egy dimenziója reprezentál, így a vektorok merőlegesek egymásra. Az algoritmus legfőbb gyengesége, hogy képtelen relációs információkat és szemantikát kódolni, így nem tudja kezelni a szinonímákat, teljesen különböző szavaknak tekinti azokat.

**Megjegyzés.** *A one-hot kódolás ritka reprezentációt eredményez.*

## Szóvektorok - folytatás

Ha egy gyors megoldásra lenne szükségünk, vagy egyszerűen szeretnénk neurális modellünk bemenetére juttatni a szöveg szavait a *one-hot* kódolás jó választás lehet. Azonban ha jelentéstartalmat szeretnénk modellezni, ennél komplexebb reprezentációs módszerre lesz szükségünk, ilyen lehet például a szóbeágyazás.

A szóbeágyazás azon a feltevésen alapszik, hogy a hasonló kontextusban előforduló szavak hasonló jelentéstartalommal bírnak. A Word2Vec és a GloVe algoritmusok képesek feldolgozni ezen relációs információt a lexémák között.

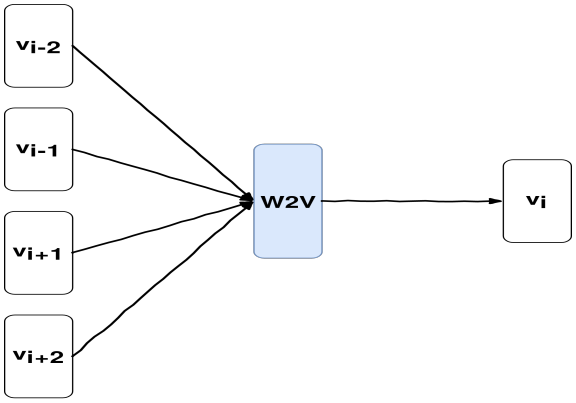
### Word2vec

A Word2Vec módszer egy sekély neurális hálón alapuló szóbeágyazási algoritmus, melyet 2013-ban mutattak be. A háló tanítását a szerzők alapvetően két felügyelet nélküli feladattal végezték: Continuous Bag of Words (CBOW) vagy Skip-Gram.

A tanítás során a mondatokat token-ekre bontották és one-hot kódolták. Majd a szöveg minden egyes token-jén végigiterálva a következőket hajtották végre:

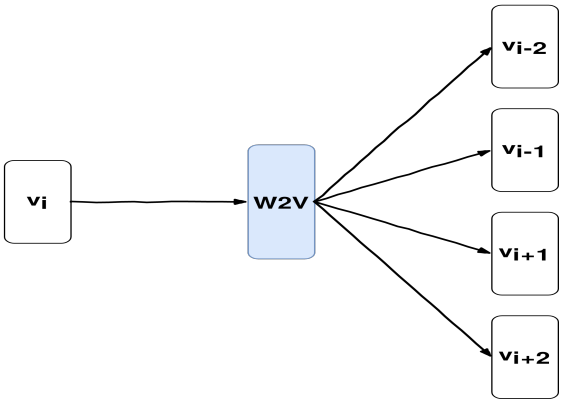
A CBOW modell szerint a háló bemenete $v_i$ $(i \in |D|)$ vektorra a $v_i$ vektor k méretű kontextusa $(v_{i-k}, ..., v_{i-1}, v_{i+1}, ..., v_{i+k} : k \in \mathbb{N})$, azaz a környezetében lévő

vektorok. A háló feladata prediktálni $v_i$ vektort a kontextus függvényében. A folyamat közben a háló rejtett rétegében létrejön a Word2Vec reprezentáció.




2.1. ábra. CBOW modell

Skip-Gram modell esetén pont az ellenkezője történik. A háló bemenete $v_i$ ($i \in |D|$) vektor lesz. A tanítás célja, hogy a háló prediktálja az i. szó k méretű kontextusának one-hot kódolt vektorait $(v_{i-k}, ..., v_{i-1}, v_{i+1}, ..., v_{i+k} : k \in \mathbb{N})$, közben a háló a rejtett rétegében megtanulja a Word2Vec reprezentációt.




2.2. ábra. Skip-Gram modell

Egy jól tanított Word2Vec modell a hasonló jelentéstartalmú szóvektorokat közel re képezi egymáshoz a vektortérben. A teljesítmény növelése érdekében finomhangolhatjuk a tanítási paramétereket. Ilyen beavatkozás lehet ha növeljük a halmaz méretét, amellyel Word2Vec modellünket tanítjuk, vagy emeljük a kontextus ablak méretét és a reprezentációs dimenziót.

**Megjegyzés.** *A Skip-Gram modell a ritka szavak, míg a CBOW modell a gyakori szavak esetén készít pontosabb reprezentációt.*

### GloVe

A Word2Vec bemutatását követő évben újabb nagy lépés történt a szóbeágyazás világában, a *Stanford University* NLP kutatócsoportja publikálta a GloVe módszert.

A GloVe (*Global Vectors*) reprezentációs módszer a Word2Vec-hez képest egy korpusz lokális statisztkáján kívül a globális statisztikáit is figyelembe veszi.

**3. Definíció.** *Adott egy korpusz, melynek elemszáma V. Az* $X \in \mathbb{N}^{V \times V}$ *mátrixot közös előfordulási mátrixnak nevezzük, ahol* $X_{ij}$ *az a szám, ahányszor i. szó kontextusában j. szó megjelenik.*

A GloVe modell tanítása egy korpusz közös előfordulási mátrixának nemnulla elemein történik. A GloVe modell egy log-bilineáris modell, amely feladata, hogy kiszámítsa a következő szó valószínűségét azon kontextusa alapján.

A módszer mögötti intuíció az, hogy a közös előfordulási valószínűségek hányadosa értékes információval szolgálhat a leképezés során. Így a feladat célja, hogy a tanult szóvektorok skaláris szorzata megegyezzen a szavak közös előfordulási valószínűségének logaritmusával. Mivel $\log\left(\frac{A}{B}\right) = \log(A) - \log(B)$, így ez a cél összekapcsolja az előfordulási valószínűségek arányszámát a vektorok távolságával.

Ugyan a globális statisztikáknak köszönhetően a GloVe több feladatban is túlteljesíti a Word2Vec-et, a tanításához szükséges közös előfordulási mátrix memóriaigénye magas. Paraméterhangolás esetén újból fel kell építenie a mátrixot, mely költséges művelet.

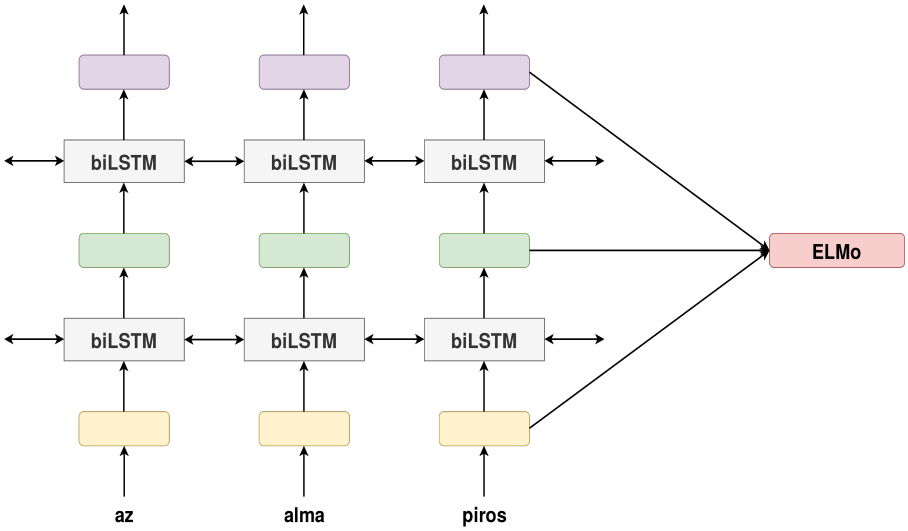
### ELMo

A Word2Vec és a GloVe már képes szemantikus információ leképezésére, azonban esetükben az ellentétes szópárok közel kerülnek egymáshoz. Azon feladatoknál, melyeknél az ellentétes szavak kiemelt szerepűek – például a hangulatelemzés – limitációk jelentkezhetnek, továbbá ezen algoritmusok rosszul kezelik az ismeretlen szavakat is.

Míg a Word2Vec és a GloVe csak szavankénti kontextusfüggetlen reprezentáció tanulására képes – azaz nem számít az adott szó környezete, melyre alkalmazzák –, addig az Embeddings from Language Models (ELMo) figyelembe veszi a lexémák

kontextusát, mondaton belüli elhelyezkedését is. Az ELMo használat közben állítja elő a vektorokat.

A modell tanításához használt neurális háló több réteg kétirányú LSTM (biLSTM) konkatenációja. A különböző rétegek más és más típusú információt képesek eltárolni.




2.3. ábra. ELMo modell

Az ELMo a különböző rétegek kimenetének feladatspecifikus kombinációján alapszik. Egy adott NLP feladatra minden biLSTM réteg egyedi súlyt kap. A végső háló 2 darab biLSTM rétegből áll, minden LSTM réteg 4096 széles.

Az így kapott sekély kétirányú módszer jelentősen javított a szóvektorok pontosságán.

Bár az ELMo egy karakter (konkatenáció) alapú reprezentációs algoritmus, szavakat ábrázol. Ezen tulajdonsága alapján képes kezelni az addig nem látott szavak problémáját is.

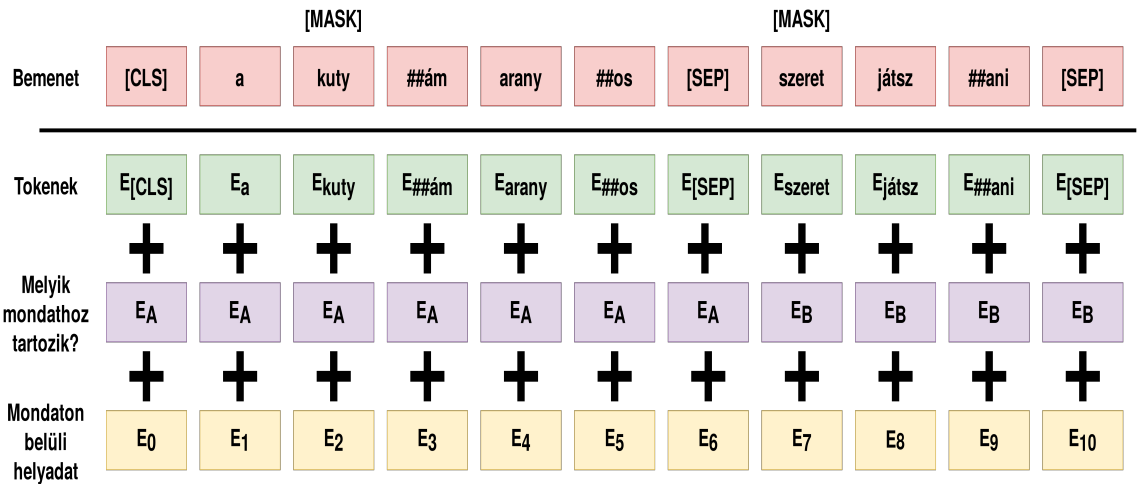
### BERT

Egy 2018-ban publikált cikk rámutatott arra, hogy a karakteralapú algoritmusok nem teljesítenek olyan jól, mint a szóalapú társaik. A Bidirectional Encoder Representations from Transformers (BERT) egy a Google által kifejlesztett transformer architektúrájú nyelvi modell. Az ELMo-hoz hasonlóan ez is kétirányú, azaz

egy szó mindkét oldali kontextusát figyelembe veszi a tanulás alatt. A BERT azonban bemenetként nem szavakat és nem is karaktereket kap, hanem szótöredékeket.

A tanítást a *transfer learning* szerint két fázisra bontották: előtanítás és finomhangolás.

Az előtanítás két feladatból állt: *következő mondat* és *maszkolás*. A bemenetben megadták a szótöredék token-eket, a token-ekhez tartozó mondaton belüli helyadatokat és azt, hogy az adott token A vagy B mondat közül melyikhez tartozik.




2.4. ábra. A BERT bemenete

A *következő mondat* esetében a mélyháló feladata kitalálni, hogy A[SEP]B input mondatokra B rákövetkezője-e A-nak. A *maszkolás* során véletlenszerűen letakarták a token-eket a mondatokban és a mélyháló megpróbálta kitalálni, hogy eredetileg melyik szó volt a [MASK] token helyén. A [CLS] token a klasszifikációs feladat alatt a mondatot ábrázolja, a [SEP] a mondatok közötti szeparátor és a [MASK] a letakart szótöredékeket helyettesíti.

A továbbiakban a modell finomhangolása az adott NLP feladat szerint történik.

Míg az ELMo különböző balról-jobbra és jobbról-balra olvasó rétegek konkatenációjaként állítja elő a vektorokat, addig a BERT a valódi mély architektúrájával csak egyszer dolgozza fel a token-eket. A *transformer* architektúra nem igényel vektoriális bemenetet, saját reprezentációt épít a token-ek számára is.

A BERT szótöredék alapú megoldása egyesíti a karakteralapú modellek előnyét a szóalapú modellek előnyével. Képes kezelni az ismeretlen szavakat és performanciája mégis magas marad. A *következő mondat* feladat a szövegben található mondatok

közötti relációk, a *maszkolás* pedig a mondatokon belüli szemantikai és szintaktikai információ ábrázolását segíti. Több NLP feladat megoldásában is jelenleg a BERT a *State-of-the-art*.

## 2.2. Reprezentáció a mondatok és magasabb nyelvi elemek szintjén

Ahogy a lexémák szemantikai tartalmát sem határozza meg az őket alkotó karakterek lánca, úgy a mondatok sem értelmezhetőek pusztán a magukban foglalt szavak halmaza alapján. A mondatok és magasabb nyelvi elemek interpretálása során fontos tényezők lehetnek a bennük lévő szintaktikai viszonyok és a kontextus is.

Néhány NLP feladatnál, mint például a dokumentumok szemantikus keresésénél, vagy szöveg összegzésnél szükség lehet magasabb szintű reprezentációkra. Ezek a módszerek szavak helyett mondatokat, bekezdéseket, vagy akár egész dokumentumokat tesznek numerikusan értelmezhetővé.

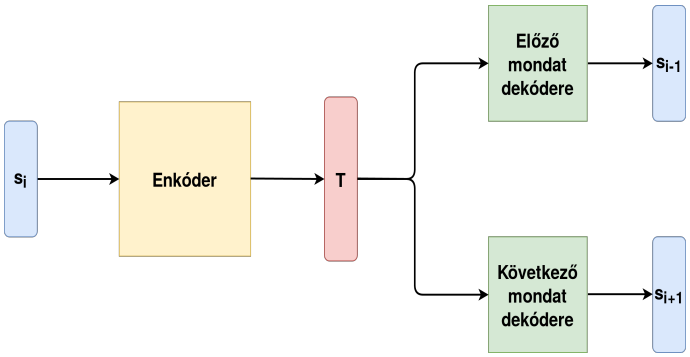
### 2.2.1. Mondatvektorok

A szóvektorokhoz hasonlóan úgy kaphatunk mondatvektorokat, ha mondatokat helyezünk el egy vektortérbe. A tanítás során azonban a sorrendiség, az egyes lexémák változó súlya és a szintaktikai viszonyok megnehezíthetik dolgunkat. Szükség van egy technikára, mely segítségével leképezhetjük és szemantikai tartalmuknál fogva összegezhetjük a megfelelő rendezett szóvektorok sorozatát, így hozzájutva az adott mondat reprezentációjához. A módszerünk akkor hatékony, ha az azonos jelentéstartalmú mondatvektorok klaszterekbe tömörülnek a vektortérben.

#### Skip-thought vektorok

A Skip-though egy 2015-ben bemutatott mondatreprezentációs módszer – a Skip-Gram algoritmus kiterjesztése – , amely a környező mondatokat is figyelembe veszi a tanulás során.

A szerzők rekurrens enkóder-dekóder architektúrát használtak a tanításhoz. A neurális háló bemenete mondathármasok szavainak Word2Vec vektoraiból állt. A háló feladata $s_i$ mondat esetén $s_{i-1}$ és $s_{i+1}$ mondatok generálása volt.




2.5. ábra. A Skip-thought enkóder-dekóder architektúrája

Az enkóder és dekóder blokkokhoz rekurrens hálót használtak, melyek lehetnek LSTM és GRU rétegek is. Az enkóder célja a legjobb teljesítményével segíteni a dekóder blokkokat, míg a dekóder blokkok célja minimalizálni az előző és a következő mondat rekonstrukciós hibáját.

Olyan szavak esetén, melyeket a háló még nem ismert, tanítottak egy $f : V_{w2v} \rightarrow V_{rnn}$ lineáris leképezést, ahol $V_{w2v}$ és $V_{rnn}$ rendre a Word2Vec és a rekurrens modell szótára. A reprezentáció vektora a rejtett, úgy nevezett *thought* vektor (T).

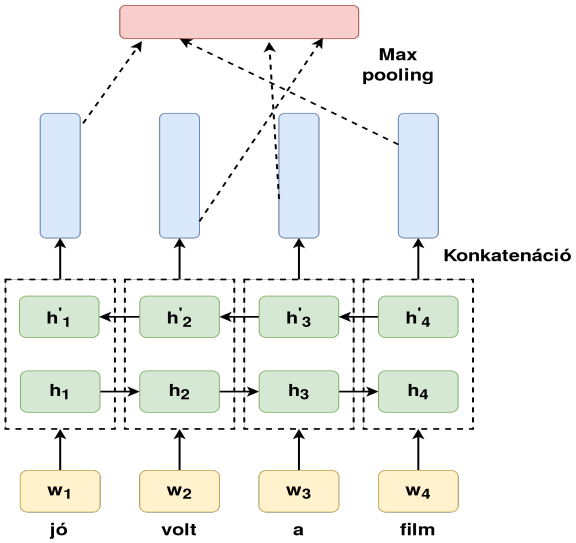
Bár a *Skip-thought* módszer képes a mondaton belüli és kívüli sorrendiségi információ leképezésére is, csak olyan esetben teljesít megfelelően, ahol az egyes mondatok – melyekre alkalmazzák – megfelelő kontextusban szerepelnek, nem izoláltak.

### InferSent

2017-ben a Facebook kutatói jelentős áttörést értek el a mondatszintű reprezentációs módszerek terén, a technika neve InferSent. Hasonló algoritmusokkal ellentétben a szerzők felügyelt tanítást végeztek, melyhez az SNLI adathalmazt vették igénybe. A cikk megmutatta, hogy egy kisebb adathalmazon történő felügyelt tanítás felülmúlhatja a nagyobb adathalmazon nem felügyelt módon tanított modellek teljesítményét.

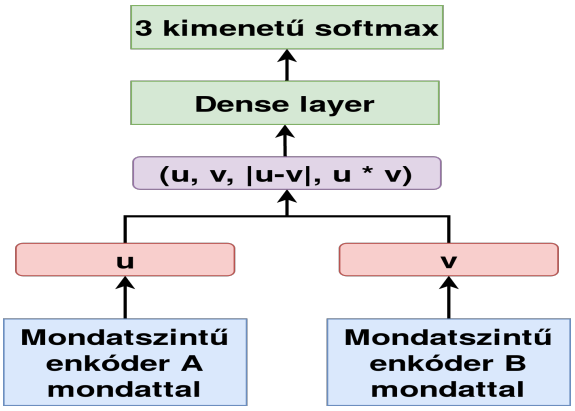
Az SNLI adathalmaz 570 ezer darab – ember által írt és címkézett – mondatpárból áll. A címkék a következők: következmény, ellentmondás és semleges.

Négyféle neurális architektúrát összemérve a legpontosabb eredményt a biLSTM + max pooling mutatta.




2.6. ábra. A biLSTM + max pooling architektúra

Az SNLI feldolgozásához szükséges NLI feladat speciális szerkezetet igényel. Mivel kontextusfüggetlen reprezentációt akartak előállítani, amely izolált formában is működik, a mondatpárok GloVe vektorait szeparáltan enkódolták.




2.7. ábra. Az NLI feladat

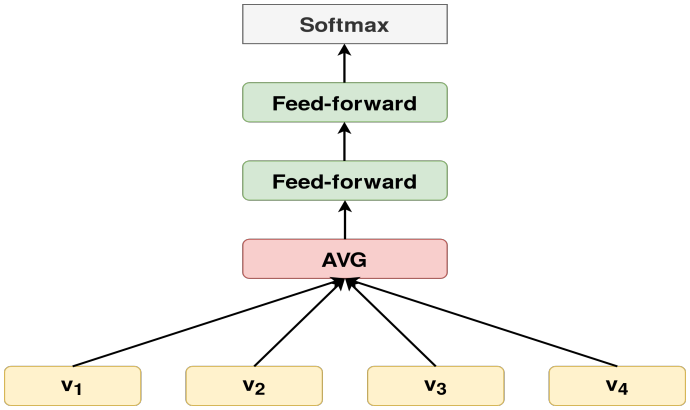
Az így készült u és v vektorokból egy speciális reprezentáció készült: u, v, $|u - v|$ és $u * v$ (vektoriális szorzat) konkatenációjával, melyet végül egy 3 osztályú klasszifikáló hálóba vezettek.

A szerzők a reprezentációs vektorméret növelésével pontosabb eredményt kaptak, de a vektorok memóriaigénye emelkedett. Az InferSent megoldja a kontextusfüggőség problémáját, így a módszer már szövegrészletekre is alkalmazható.

**Megjegyzés.** *Az InferSent napjaink egyik legjobb teljesítményű szemantikus reprezentációs algoritmusa.*

### USE

Az InferSent bemutatását követő évben a Google Research csapata a modern reprezentációs módszereket vizsgálta a *transfer learning* aspektusából. A Universal Sentence Encoder (USE) egy mondatszintű algoritmus, mely célja, hogy a használója könnyedén igényeire tudja formálni, annak érdekében, hogy pontosabb leképezést kapjon. A szerzők két architektúrát használtak: a BERT-ben említett *transformer*-t és a DAN-t (*Deep Averaging Network*).




2.8. ábra. DAN architektúra

A *transformer* modell egyik algráfja a mondatokban lévő szavak kontextusfüggő reprezentációját állítja elő. A folyamat során figyelembe veszi az egyes lexémák sorrendi és egyéni információit is, majd összegzi őket, így megkapja a végső mondatszintű reprezentációt. A *DAN* modell az input tokenek vektorait először átlagolja, majd *feed forward* rétegek segítségével előállítja a mondatvektorokat. A USE szavakból, mondatokból, vagy akár rövidebb bekezdésekből is képes 512 méretű vektorokat generálni.

A neurális hálók tanítását két részre bontották, bemenetként angol nyelvű karakterláncokat kaptak. Az első rész a *Skip-thought*-hoz hasonló módon, dialógusokból vett mondat-válasz párokkal, illetve felügyelt módon a *Stanford Natural Language Inference* (SNLI) korpuszon történt.

A cikk során kiemelt szerepet kapott a tanítás második fázisa. Számos módon finomhangolták a modelleket és mérték a teljesítményüket. A feladatok közé tarto-

zott, hogy filmes értékelések szövege alapján ki kellett találnia a neurális hálóknak az értékelések pontszámát 1 és 5 között. Továbbá vásárlói értékelések hangulati töltetét kellett prediktálniuk.

Az algoritmusokat kipróbálták a szavak szintjén, a mondatok szintjén és a kettő módszer konkatenációjaként is. A legjobb teljesítményt a mondatszintű reprezentációk mutatták. A transformer architektúra pontosabb eredményt hozott, mint a DAN alapú modell, de a transformer modell $\mathcal{O}(n^2)$, míg a DAN modell $\mathcal{O}(n)$ időkomplexitású a bemeneti hossz függvényében. Továbbá memóriahasználatban is kedvezőbb választás a DAN.

A *GloVe*-hoz hasonlóan a USE is képes asszociációkra, de jóval gyengébb ezen képessége az olyan kényes témák esetében, mint a szexizmus és a rasszizmus. Ez a tény alkalmassá teheti a USE-t az ipari használatra is.

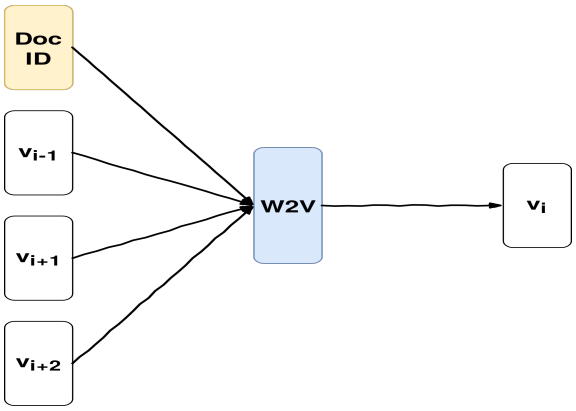
A szerzők rávilágítottak arra, hogy kevés adat esetén jó választás lehet a *transfer learning* módszere, és a magasabb szintű reprezentációk pontosabb eredményt érhetnek el a legtöbb feladat esetében.

### 2.2.2. Dokumentumszintű reprezentáció

Ahogy a technológia fejlődik, úgy növekszik a világon az egységnyi idő alatt előállított információ mennyisége is. Gyakori eset, hogy ez írott formában, dokumentumokban jelenik meg. Dokumentumnak tekinthetünk minden, a mondatnál hosszabb emberi nyelven írott szöveget.

Bár a magasabb nyelvi egységek értelmezése és feldolgozása sok területen előkerülő feladat, mégsem triviális. Nagy kihívást jelent a szemantikai szimilaritás mérése, az olyan gyakorlati problémákat nem is említve, mint a duplikációk kiszűrése a fórumokról, vagy a szociális média analízis.

2014-ben a Word2Vec szerzői előálltak egy dokumentumszintű reprezentációs algoritmussal. A Doc2Vec módszer a Word2Vec modell kiterjesztése a dokumentumok szintjére. Mivel a dokumentumokat nem lehet a szavakhoz hasonló logikai struktúrába rendezni, ezért a megszokott *CBOW* modell bemeneti vektorai mellé egy speciális, a magasabb nyelvi elem azonosítóját jelölő vektort konkatenáltak. Az algoritmus neve PV-DM. A modell tanítása végén a speciális vektor reprezentációja képviseli a dokumentumot.




2.9. ábra. Doc2Vec PV-DM architektúra

A Word2Vec-hez hasonlóan a Doc2Vec-nek is létezik *Skip-Gram* alternatívája, ez a PV-DBOW. A PV-DBOW modell gyorsabb és memóriahasználat szempontjából is gazdaságosabb a PV-DM-hez képest.

Mivel relatíve kevés algoritmus képes dokumentumszintű modellezésre és azok teljesítménye is limitált, a Doc2Vec egy jó választás lehet. A modell egyszerre mutat jó teljesítményt és a használata is könnyű.

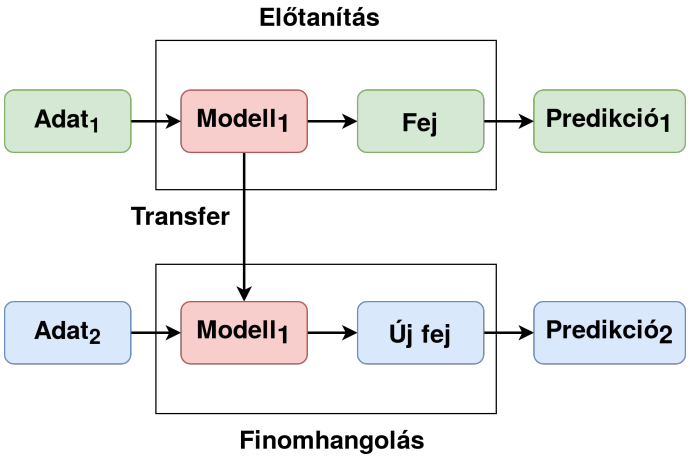
## 2.3. Transfer learning

A modern szemantikus reprezentációs algoritmusok tanítása összetett folyamat. A feladatok során egyszerre kell több szempontra figyelni, melyek befolyásolhatják a modellünk pontosságát. Példának okáért mondatszintű reprezentációnknak képesnek kell lennie értelmezni a lexémák egymáshoz fűződő viszonyait és a mondatok közötti kohéziót is. A *transfer learning* egy kiváló eszköz arra, hogy modellünket tanítsuk több aspektus szerint.

A *transfer learning* napjainkban közkedvelt tanítási módszer, melynek ötletét az NLP ágazata a számítógépes látás eszközkészletéből merítette. A folyamatot két fázisra lehet bontani: előtanítás és a finomhangolás. Az előtanítás általában nagy mennyiségű adaton történik. A finomhangolás az előtanítás után kapott modell – adott NLP feladathoz szükséges – speciális feladatokon való tanítását jelenti, amely szignifikánsan kevesebb adatot igényel.

**4. Definíció.** *Jelölje $D_s$ a forrástartományt, $D_t$ a céltartományt, $T_s$ a forrástartományhoz tartozó feladatot, továbbá $X_t$ és $Y_t$ rendre a $T_t$ célfeladathoz tartozó in-*

*putváltozók és címkék halmazát. A* ***transfer learning*** *célja megtanulni* $P(Y_t|X_t)$ *feltételes eloszlást* $D_t$*-ben* $D_s$ *által gyűjtött információ alapján úgy, hogy* $D_s \neq D_t$ *vagy* $T_s \neq T_t$*.*




2.10. ábra. Transfer learning

Az előtanítási szakasz olyan feladattal kezdődik, mely kellőképpen generalizál és a neurális hálónk sok, hasznos és általános információhoz tud jutni. A folyamathoz használt adathalmaz általában nagy mennyiségű annotálatlan adatot tartalmaz, de vannak kivételek, például az InferSent esetében.

A finomhangolási fázis alatt használt feladatok az előtanítás után kapott modell súlyait alkalmazzák, de a bemeneti adatok és a feladatok végrehajtásához szükséges fej lehet eltérő is. Ezen szakasz állhat feladatok sorozatából is, ekkor a súlyokat inkrementálisan használják azok. A sikeres végrehajtást követően modellünk képes lesz komplexebb összefüggések felismerésére és pontosabb eredmény elérésére.

A jelenlegi trendek szerint a reprezentációs módszerek tanítási módja nagyobb hangsúlyt kap, mint maga a neurális háló szerkezete. A *transfer learning* használata a numerikus ábrázolás során még kiaknázatlan terület, mely rendkívül sok eredményt hozhat a jövőben.

# 3. fejezet

# Felhasználói dokumentáció

Lorem ipsum dolor sit amet $\mathbb{N}$, consectetur adipiscing elit. Duis nibh leo, dapibus in elementum nec, aliquet id sem. Suspendisse potenti. Nullam sit amet consectetur nibh. Donec scelerisque varius turpis at tincidunt. Cras a diam in mauris viverra vehicula. Vivamus mi odio, fermentum vel arcu efficitur, lacinia viverra nibh. Aliquam aliquam ante mi, vel pretium arcu dapibus eu. Nulla finibus ante vel arcu tincidunt, ut consectetur ligula finibus. Mauris mollis lectus sed ipsum bibendum, ac ultrices erat dictum. Suspendisse faucibus euismod lacinia $\mathbb{Z}$.

## 3.1. Felsorolások

Etiam vel odio ante. Etiam pulvinar nibh quis massa auctor congue. Pellentesque quis odio vitae sapien molestie vestibulum sit amet et quam. Pellentesque vel dui eget enim hendrerit finibus at sit amet libero. Quisque sollicitudin ultrices enim, nec porta magna imperdiet vitae. Cras condimentum nunc dui, eget molestie nunc accumsan vel.

- Fusce in aliquet neque, in pretium sem.
- Donec tincidunt tellus id lectus pretium fringilla.
- Nunc faucibus, erat pretium tempus tempor, tortor mi fringilla neque, ac congue ex dui vitae mauris.

Donec dapibus sodales ante, at scelerisque nunc laoreet sit amet. Mauris porttitor tincidunt neque, vel ullamcorper neque pulvinar et. Integer eu lorem euismod,

faucibus lectus sed, accumsan felis. Nunc ornare mi at augue vulputate, eu venenatis magna mollis. Nunc sed posuere dui, et varius nulla. Sed mollis nibh augue, eget scelerisque eros ornare nec.

1. Donec pretium et quam a cursus. Ut sollicitudin tempus urna et mollis.
2. Aliquam et aliquam turpis, sed fermentum mauris. Nulla eget ex diam.
3. Donec eget tellus pharetra, semper neque eget, rutrum diam Step 1.

Praesent porta, metus eget eleifend consequat, eros ligula eleifend ex, a pellentesque mi est vitae urna. Vivamus turpis nunc, iaculis non leo eget, mattis vulputate tellus. Maecenas rutrum eros sem, pharetra interdum nulla porttitor sit amet. In vitae viverra ante. Maecenas sit amet placerat orci, sed tincidunt velit. Vivamus mattis, enim vel suscipit elementum, quam odio venenatis elit[2], et mollis nulla nunc a risus. Praesent purus magna, tristique sed lacus sit amet, convallis malesuada magna.

**Vestibulum venenatis** malesuada enim, ac auctor erat vestibulum et. Phasellus id purus a leo suscipit accumsan.

**Orci varius natoque** penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam interdum rhoncus nisl, vel pharetra arcu euismod sagittis. Vestibulum ac turpis auctor, viverra turpis at, tempus tellus.

**Morbi dignissim** erat ut rutrum aliquet. Nulla eu rutrum urna. Integer non urna at mauris scelerisque rutrum sed non turpis.

### 3.1.1. Szoros térközű felsorolások

Phasellus ultricies, sapien sit amet ultricies placerat, velit purus viverra ligula, id consequat ipsum odio imperdiet enim:

1. Maecenas eget lobortis leo.
2. Donec eget libero enim.
3. In eu eros a eros lacinia maximus ullamcorper eget augue.

[2]Phasellus faucibus varius purus, nec tristique enim porta vitae.

In quis turpis metus. Proin maximus nibh et massa eleifend, a feugiat augue porta. Sed eget est purus. Duis in placerat leo. Donec pharetra eros nec enim convallis:

- Pellentesque odio lacus.
- Maximus ut nisl auctor.
- Sagittis vulputate lorem.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Sed lorem libero, dignissim vitae gravida a, ornare vitae est.

**Cras maximus** massa commodo pellentesque viverra.

**Morbi sit** amet ante risus. Aliquam nec sollicitudin mauris

**Ut aliquam rhoncus sapien** luctus viverra arcu iaculis posuere

## 3.2. Képek, ábrák

Aliquam vehicula luctus mi a pretium. Nulla quam neque, maximus nec velit in, aliquam mollis tortor. Aliquam erat volutpat. Curabitur vitae laoreet turpis. Integer id diam ligula. Nulla sodales purus id mi consequat, eu venenatis odio pharetra. Cras a arcu quam. Suspendisse augue risus, pulvinar a turpis et, commodo aliquet turpis. Nulla aliquam scelerisque mi eget pharetra. Mauris sed posuere elit, ac lobortis metus. Proin lacinia sit amet diam sed auctor. Nam viverra orci id sapien sollicitudin, a aliquam lacus suscipit, Figure 3.1:


3.1. ábra. Quisque ac tincidunt leo

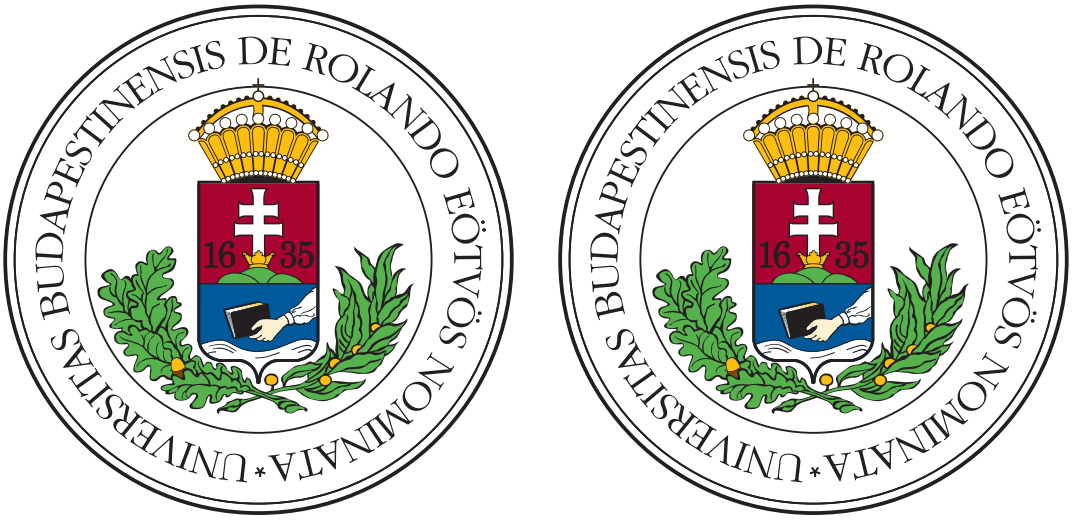
### 3.2.1. Képek szegélyezése

Ut aliquet nec neque eget fermentum. Cras volutpat tellus sed placerat elementum. Quisque neque dui, consectetur nec finibus eget, blandit id purus. Nam eget ipsum non nunc placerat interdum.


3.2. ábra. Quisque ac tincidunt leo

### 3.2.2. Képek csoportosítása

In non ipsum fermentum urna feugiat rutrum a at odio. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nulla tincidunt mattis nisl id suscipit. Sed bibendum ac felis sed volutpat. Nam pharetra nisi nec facilisis faucibus. Aenean tristique nec libero non commodo. Nulla egestas laoreet tempus. Nunc eu aliquet nulla, quis vehicula dui. Proin ac risus sodales, gravida nisi vitae, efficitur neque, Figure 3.3:




(a) Vestibulum quis mattis urna

(b) Donec hendrerit quis dui sit amet venenatis

3.3. ábra. Aenean porttitor mi volutpat massa gravida

Nam et nunc eget elit tincidunt sollicitudin. Quisque ligula ipsum, tempor vitae tortor ut, commodo rhoncus diam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Phasellus vehicula quam dui, eu convallis metus porta ac.

## 3.3. Táblázatok

Nam magna ex, euismod nec interdum sed, sagittis nec leo. Nam blandit massa bibendum mattis tristique. Phasellus tortor ligula, sodales a consectetur vitae, placerat vitae dolor. Aenean consequat in quam ac mollis.

| **Phasellus tortor** | **Aenean consequat** |
|---|---|
| *Sed malesuada* | Aliquam aliquam velit in convallis ultrices. |
| *Purus sagittis* | Quisque lobortis eros vitae urna lacinia euismod. |
| *Pellentesque* | Curabitur ac lacus pellentesque, eleifend sem ut, placerat enim. Ut auctor tempor odio ut dapibus. |

3.1. táblázat. Maecenas tincidunt non justo quis accumsan

### 3.3.1. Sorok és oszlopok egyesítése

Mauris a dapibus lectus. Vestibulum commodo nibh ante, ut maximus magna eleifend vel. Integer vehicula elit non lacus lacinia, vitae porttitor dolor ultrices. Vivamus gravida faucibus efficitur. Ut non erat quis arcu vehicula lacinia. Nulla felis mauris, laoreet sed malesuada in, euismod et lacus. Aenean at finibus ipsum. Pellentesque dignissim elit sit amet lacus congue vulputate.

| **Quisque** | **Suspendisse** | | **Aliquam** | | **Vivamus** | |
|---|---|---|---|---|---|---|
| | Proin | Nunc | Proin | Nunc | Proin | Nunc |
| Leo | 2,80 MB | 100% | 232 KB | 8,09% | 248 KB | 8,64% |
| Vel | 9,60 MB | 100% | 564 KB | 5,74% | 292 KB | 2,97% |
| Auge | 78,2 MB | 100% | 52,3 MB | 66,88% | 3,22 MB | 4,12% |

3.2. táblázat. Vivamus ac arcu fringilla, fermentum neque sed, interdum erat. Mauris bibendum mauris vitae enim mollis, et eleifend turpis aliquet.

### 3.3.2. Több oldalra átnyúló táblázatok

Nunc porta placerat leo, sit amet porttitor dui porta molestie. Aliquam at fermentum mi. Maecenas vitae lorem at leo tincidunt volutpat at nec tortor. Vivamus

semper lacus eu diam laoreet congue. Vivamus in ipsum risus. Nulla ullamcorper finibus mauris non aliquet. Vivamus elementum rhoncus ex ut porttitor.

| **Praesent aliquam mauris enim** | |
|---|---|
| *Suspendisse potenti* | *Lorem ipsum dolor sit amet* |
| *Praesent* | Nulla ultrices et libero sit amet fringilla. Nunc scelerisque ante tempus sapien placerat convallis. |
| *Luctus* | Integer hendrerit erat massa, non hendrerit risus convallis at. Curabitur ultrices, justo in imperdiet condimentum, neque tortor luctus enim, luctus posuere massa erat vitae nibh. |
| *Egestas* | Duis fermentum feugiat augue in blandit. Mauris a tempor felis. Pellentesque ultricies tristique dignissim. Pellentesque aliquam semper tristique. Nam nec egestas dolor. Vestibulum id elit quis enim fringilla tempor eu a mauris. Aliquam vitae lacus tellus. Phasellus mauris lectus, aliquam id leo eget, auctor dapibus magna. Fusce lacinia felis ac elit luctus luctus. |
| *Dignissim* | Praesent aliquam mauris enim, vestibulum posuere massa facilisis in. Suspendisse potenti. Nam quam purus, rutrum eu augue ut, varius vehicula tellus. Fusce dui diam, aliquet sit amet eros at, sollicitudin facilisis quam. Phasellus tempor metus vel augue gravida pretium. Proin aliquam aliquam blandit. Nulla id tempus mi. Fusce in aliquam tortor. |
| *Pellentesque* | Donec felis nibh, imperdiet a arcu non, vehicula gravida nibh. Quisque interdum sapien eu massa commodo, ac elementum felis faucibus. |

| *Suspendisse potenti* | *Lorem ipsum dolor sit amet* |
|---|---|
| *Molestie* | Cras ullamcorper tellus et auctor ultricies. Maecenas tincidunt euismod lectus nec venenatis. Suspendisse potenti. Pellentesque pretium nunc ut euismod cursus. Nam venenatis condimentum quam. Curabitur suscipit efficitur aliquet. Interdum et malesuada fames ac ante ipsum primis in faucibus. |
| *Vivamus semper* | In purus purus, faucibus eu libero vulputate, tristique sodales nunc. Nulla ut gravida dolor. Fusce vel pellentesque mi, vel efficitur eros. Nunc vitae elit tellus. Sed vestibulum auctor consequat. |
| *Condimentum* | Nulla scelerisque, leo et facilisis pretium, risus enim cursus turpis, eu suscipit ipsum ipsum in mauris. Praesent eget pulvinar ipsum, suscipit interdum nunc. Nam varius massa ut justo ullamcorper sollicitudin. Vivamus facilisis suscipit neque, eu fermentum risus. Ut at mi mauris. |

3.3. táblázat. Praesent ullamcorper consequat tellus ut eleifend

# 4. fejezet

# Fejlesztői dokumentáció

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis nibh leo, dapibus in elementum nec, aliquet id sem. Suspendisse potenti. Nullam sit amet consectetur nibh. Donec scelerisque varius turpis at tincidunt.

## 4.1. Tételek, definíciók, megjegyzések

**5. Definíció.** *Mauris tristique sollicitudin ultrices. Etiam tristique quam sit amet metus dictum imperdiet. Nunc id lorem sed nisl pulvinar aliquet vitae quis arcu. Morbi iaculis eleifend porttitor.*

Maecenas rutrum eros sem, pharetra interdum nulla porttitor sit amet. In vitae viverra ante. Maecenas sit amet placerat orci, sed tincidunt velit. Vivamus mattis, enim vel suscipit elementum, quam odio venenatis elit, et mollis nulla nunc a risus. Praesent purus magna, tristique sed lacus sit amet, convallis malesuada magna. Phasellus faucibus varius purus, nec tristique enim porta vitae.

**1. Tétel.** *Nulla finibus ante vel arcu tincidunt, ut consectetur ligula finibus. Mauris mollis lectus sed ipsum bibendum, ac ultrices erat dictum. Suspendisse faucibus euismod lacinia. Etiam vel odio ante.*

*Bizonyítás.* Etiam pulvinar nibh quis massa auctor congue. Pellentesque quis odio vitae sapien molestie vestibulum sit amet et quam. Pellentesque vel dui eget enim hendrerit finibus at sit amet libero. Quisque sollicitudin ultrices enim, nec porta magna imperdiet vitae. Cras condimentum nunc dui. □

Donec dapibus sodales ante, at scelerisque nunc laoreet sit amet. Mauris porttitor tincidunt neque, vel ullamcorper neque pulvinar et. Integer eu lorem euismod, faucibus lectus sed, accumsan felis.

**Emlékeztető.** *Nunc ornare mi at augue vulputate, eu venenatis magna mollis. Nunc sed posuere dui, et varius nulla. Sed mollis nibh augue, eget scelerisque eros ornare nec. Praesent porta, metus eget eleifend consequat, eros ligula eleifend ex, a pellentesque mi est vitae urna. Vivamus turpis nunc, iaculis non leo eget, mattis vulputate tellus.*

Fusce in aliquet neque, in pretium sem. Donec tincidunt tellus id lectus pretium fringilla. Nunc faucibus, erat pretium tempus tempor, tortor mi fringilla neque, ac congue ex dui vitae mauris. Donec pretium et quam a cursus.

**Megjegyzés.** *Aliquam vehicula luctus mi a pretium. Nulla quam neque, maximus nec velit in, aliquam mollis tortor. Aliquam erat volutpat. Curabitur vitae laoreet turpis. Integer id diam ligula.*

Ut sollicitudin tempus urna et mollis. Aliquam et aliquam turpis, sed fermentum mauris. Nulla eget ex diam. Donec eget tellus pharetra, semper neque eget, rutrum diam.

### 4.1.1. Egyenletek, matematika

Duis suscipit ipsum nec urna blandit, $2 + 2 = 4$ pellentesque vehicula quam fringilla. Vivamus euismod, lectus sit amet euismod viverra, dolor metus consequat sapien, ut hendrerit nisl nulla id nisi. Nam in leo eu quam sollicitudin semper a quis velit.

$$a^2 + b^2 = c^2$$

Phasellus mollis, elit sed convallis feugiat, dolor quam dapibus nibh, suscipit consectetur lacus risus quis sem. Vivamus scelerisque porta odio, vitae euismod dolor accumsan ut.

In mathematica, identitatem Euleri (equation est scriptor vti etiam notum) sit aequalitatem Equation 4.1:

$$e^{i \times \pi} + 1 = 0 \tag{4.1}$$

## 4.2. Forráskódok

Nulla sodales purus id mi consequat, eu venenatis odio pharetra. Cras a arcu quam. Suspendisse augue risus, pulvinar a turpis et, commodo aliquet turpis. Nulla aliquam scelerisque mi eget pharetra. Mauris sed posuere elit, ac lobortis metus. Proin lacinia sit amet diam sed auctor. Nam viverra orci id sapien sollicitudin, a aliquam lacus suscipit. Quisque ac tincidunt leo Code 4.1 and 4.2:

```
#include <stdio>

int main()
{
  int c;
  std::cout << "Hello World!" << std::endl;

  std::cout << "Press any key to exit." << std::endl;
  std::cin >> c;

  return 0;
}
```

4.1. forráskód. Hello World in C++

```
using System;
namespace HelloWorld
{
  class Hello
  {
    static void Main()
    {
      Console.WriteLine("Hello World!");

      Console.WriteLine("Press any key to exit.");
      Console.ReadKey();
    }
  }
}
```

4.2. forráskód. Hello World in C#

### 4.2.1. Algoritmusok

A general Interval Branch and Bound algorithm is shown in Algorithm 1. One of the following selection rules is applied in Step 3.

Példa forrása: Acta Cybernetica (ez egy link).

**Algoritmus 1** A general interval B&B algorithm

**Funct** IBB$(S, f)$

1: Set the working list $\mathcal{L}_W := \{S\}$ and the final list $\mathcal{L}_Q := \{\}$
2: **while** ( $\mathcal{L}_W \neq \emptyset$ ) **do**
3: Select an interval $X$ from $\mathcal{L}_W$ *Selection rule*
4: Compute $lbf(X)$ *Bounding rule*
5: **if** $X$ cannot be eliminated **then** *Elimination rule*
6: Divide $X$ into $X^j,\ j = 1, \ldots, p$, subintervals *Division rule*
7: **for** $j = 1, \ldots, p$ **do**
8: **if** $X^j$ satisfies the termination criterion **then** *Termination rule*
9: Store $X^j$ in $\mathcal{L}_W$
10: **else**
11: Store $X^j$ in $\mathcal{L}_W$
12: **end if**
13: **end for**
14: **end if**
15: **end while**
16: **return** $\mathcal{L}_Q$

# 5. fejezet

# Összegzés

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In eu egestas mauris. Quisque nisl elit, varius in erat eu, dictum commodo lorem. Sed commodo libero et sem laoreet consectetur. Fusce ligula arcu, vestibulum et sodales vel, venenatis at velit. Aliquam erat volutpat. Proin condimentum accumsan velit id hendrerit. Cras egestas arcu quis felis placerat, ut sodales velit malesuada. Maecenas et turpis eu turpis placerat euismod. Maecenas a urna viverra, scelerisque nibh ut, malesuada ex.

Aliquam suscipit dignissim tempor. Praesent tortor libero, feugiat et tellus porttitor, malesuada eleifend felis. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam eleifend imperdiet lorem, sit amet imperdiet metus pellentesque vitae. Donec nec ligula urna. Aliquam bibendum tempor diam, sed lacinia eros dapibus id. Donec sed vehicula turpis. Aliquam hendrerit sed nulla vitae convallis. Etiam libero quam, pharetra ac est nec, sodales placerat augue. Praesent eu consequat purus.

# A. függelék

# Szimulációs eredmények

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque facilisis in nibh auctor molestie. Donec porta tortor mauris. Cras in lacus in purus ultricies blandit. Proin dolor erat, pulvinar posuere orci ac, eleifend ultrices libero. Donec elementum et elit a ullamcorper. Nunc tincidunt, lorem et consectetur tincidunt, ante sapien scelerisque neque, eu bibendum felis augue non est. Maecenas nibh arcu, ultrices et libero id, egestas tempus mauris. Etiam iaculis dui nec augue venenatis, fermentum posuere justo congue. Nullam sit amet porttitor sem, at porttitor augue. Proin bibendum justo at ornare efficitur. Donec tempor turpis ligula, vitae viverra felis finibus eu. Curabitur sed libero ac urna condimentum gravida. Donec tincidunt neque sit amet neque luctus auctor vel eget tortor. Integer dignissim, urna ut lobortis volutpat, justo nunc convallis diam, sit amet vulputate erat eros eu velit. Mauris porttitor dictum ante, commodo facilisis ex suscipit sed.

Sed egestas dapibus nisl, vitae fringilla justo. Donec eget condimentum lectus, molestie mattis nunc. Nulla ac faucibus dui. Nullam a congue erat. Ut accumsan sed sapien quis porttitor. Ut pellentesque, est ac posuere pulvinar, tortor mauris fermentum nulla, sit amet fringilla sapien sapien quis velit. Integer accumsan placerat lorem, eu aliquam urna consectetur eget. In ligula orci, dignissim sed consequat ac, porta at metus. Phasellus ipsum tellus, molestie ut lacus tempus, rutrum convallis elit. Suspendisse arcu orci, luctus vitae ultricies quis, bibendum sed elit. Vivamus at sem maximus leo placerat gravida semper vel mi. Etiam hendrerit sed massa ut lacinia. Morbi varius libero odio, sit amet auctor nunc interdum sit amet.

Aenean non mauris accumsan, rutrum nisi non, porttitor enim. Maecenas vel

tortor ex. Proin vulputate tellus luctus egestas fermentum. In nec lobortis risus, sit amet tincidunt purus. Nam id turpis venenatis, vehicula nisl sed, ultricies nibh. Suspendisse in libero nec nisi tempor vestibulum. Integer eu dui congue enim venenatis lobortis. Donec sed elementum nunc. Nulla facilisi. Maecenas cursus id lorem et finibus. Sed fermentum molestie erat, nec tempor lorem facilisis cursus. In vel nulla id orci fringilla facilisis. Cras non bibendum odio, ac vestibulum ex. Donec turpis urna, tincidunt ut mi eu, finibus facilisis lorem. Praesent posuere nisl nec dui accumsan, sed interdum odio malesuada.

# Ábrák jegyzéke

2.1. CBOW modell . . . . . 8
2.2. Skip-Gram modell . . . . . 8
2.3. ELMo modell . . . . . 10
2.4. A BERT bemenete . . . . . 11
2.5. A Skip-thought enkóder-dekóder architektúrája . . . . . 13
2.6. A biLSTM + max pooling architektúra . . . . . 14
2.7. Az NLI feladat . . . . . 14
2.8. DAN architektúra . . . . . 15
2.9. Doc2Vec PV-DM architektúra . . . . . 17
2.10. Transfer learning . . . . . 18

3.1. Quisque ac tincidunt leo . . . . . 21
3.2. Quisque ac tincidunt leo . . . . . 22
3.3. Aenean porttitor mi volutpat massa gravida . . . . . 22

# Táblázatok jegyzéke

3.1. Maecenas tincidunt non justo quis accumsan . . . . . 23
3.2. Rövid cím a táblázatjegyzékbe . . . . . 23
3.3. Praesent ullamcorper consequat tellus ut eleifend . . . . . 25

# Forráskódjegyzék

4.1. Hello World in C++ . . . . . 28
4.2. Hello World in C# . . . . . 28