Formális nyelvek beágyazása vektortérbe

Fraknói Ádám
Témavezetők: Kornai András, Zombori Zsolt

2023. május 26.

1. Bevezetés

A természetes nyelvi modellek (NLP) esetén jelentős fordulópontot jelentettek, amikor megjelentek a szemantikailag is jó szóbeágyazások egy vektortérbe. Kiderült, hogy egy jó beágyazás számos feladatnál nagyon hasznos, mint például fordításnál vagy szentiment elemzésnél, amire az eredeti rendszer nem volt tanítva. A beágyazások mögötti ötlet igazán egyszerű: egy szó jelentését jól reprezentálja az, hogy természetes szövegben milyen szavak veszik őt körül. Vitathatatlan, hogy a szavak és mondatok megfelelő reprezentációjuk kulcsfontosságúak a mai nyelvi modellekben. A matematikai érvelés során számos mélytanulási rendszert használnak, ahol a betanított modellekben implicit létrehozzák a matematikai objektumok beágyazását. A gyakorlatban nincs azonban bizonyíték arra, hogy ezek a beágyazások jók lennének a matematika szemantikájához. Azt sejtjük, hogy a nem megfelelő beágyazások a tételbizonyító rendszerek egy szűk keresztmetszete lehet, emiatt igyekeztünk jobban megérteni, hogy hogyan lehet egy jó beágyazást kapni matematikai objektumokon.

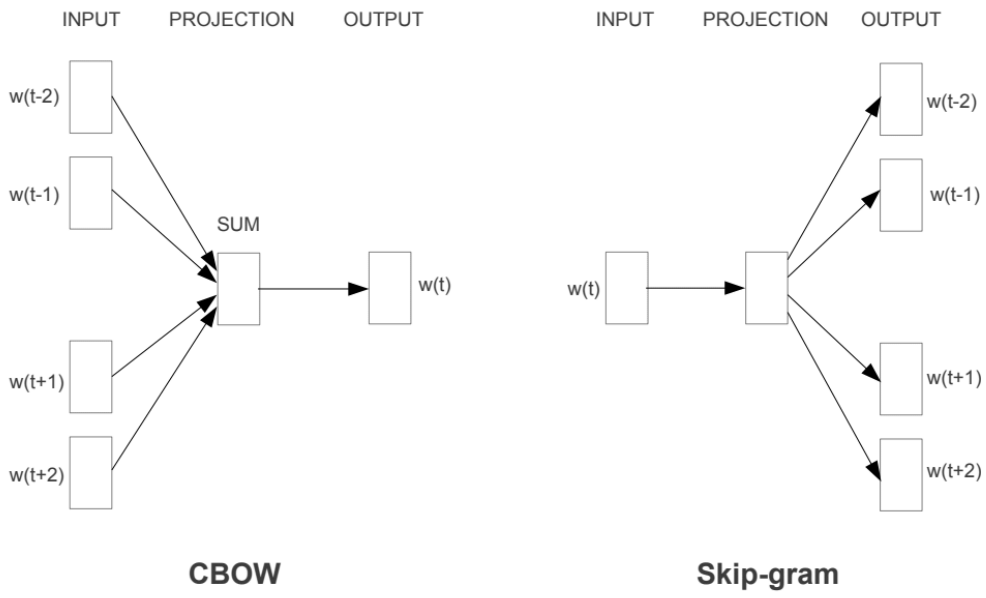
Az irodalomban több hasonló kutatás készült már e téren, például a gráf neurális hálókat (GNN) régóta különösen alkalmasnak találják matematikai képletek reprezentálására, sikerrel is alkalmazták tételbizonyítási rendszerekben [3], viszont általában a látens tér struktúrájáról nincs igazán elemzés. A kutásunk elsősorban a látens tér megértésére összpontosult szemben a modell teljesítményével különféle feladatokon.

2. Nyelvi modellek

A következőkben két algoritmust (Word2Vec [2] és BERT [1]) nézünk meg vázlatosan. A legkisebb egységek természetes nyelvi szövegek esetén a tokenek, melyek általában a szavakat reprezentálnak. A nyelvi modellek bemenetként kapnak egy szótárt (a tokenek halmazát), illetve egy szöveget (tokenek egy sorozatát), majd kimenetként pedig a szótár minden eleméhez rendelnek egy n dimenziós vektort.

2.1. Word2Vec

Két változata is van az algoritmusnak: CBOW és Skip-Gram. Az előbbiben a tanítás során a modell végigmegy minden mondaton és minden szóra megpróbálja megjósolni, hogy a környezetének függvényében milyen szó állhat ott. Az utóbbi viszont pont fordítva csinálja, a szó ismeretében próbálja meg kitalálni a környezetet.



1. ábra. Word2Vec modell architektúrája [2]

2.2. BERT

A BERT nyelvi modell az úgynevezett transformereken [4] alapul, melyek a szöveg szavai között lévő kontextuális kapcsolatokat tanulják meg. A BERT ahelyett, hogy a következő szót prediktálná, az egyik stratégiája a következő:

Véletlenszűren kiválasztja a tokenek 15%-át, aminek a 80%-át kicseréli egy speciális [MASK] tokenre, 10%-át kicseréli egy véletlenszerű tokenre, a maradék 10%-át pedig változatlanul hagyja.

A BERT előnye, hogy amellett, hogy beágyazza a tokeneket (azaz minden tokenhez hozzárendel egy n dimenziós vektort), egy tetszőleges tokensorozathoz is hozzá tud rendelni egy vektort.

3. Munkánk

3.1. Adathalmaz

Első lépésként olyan aritmetikai formulákkal kezdtünk, melyek nem tartalmaztak változókat. A struktúrájuk a következőképp nézett ki: `<exp> <rel> <exp>`, ahol `<exp>` tízes számrendszerbeli számjegyekből és az $\{+, -, *, /\}$ operátorokból állt, a `rel` pedig az egyenlőtlenségeket szimbolizálta $\{=, \leq, \geq\}$. Egy kifejezés az alábbi módon nézett ki:

$$((383 + 269)/((1 * 1) * (642 - 641))) = ((571/(391/391)) + 81)$$

Ilyen módon készítettünk különféle adathalmazokat, általában 100.000 sorral (kifejezéssel), majd ezen adathalmazokon tanítottuk be a Word2Vec, illetve BERT modelleket. Nevezzük azt az adathalmazt, amikor `<rel>` kizárólag a $\{=\}$ operátorból áll úgy, hogy *normal*, míg ha $\{=, \leq, \geq\}$ operátorokból, akkor *relation*.

Mindkét esetben egy tokennek egy karakter fog megfeleleni, azaz a tokenek a tízes számrendszerbeli számok, négy műveletjel, zárójelek, relációk, illetve néhány a BERT-hez szükséges beépített token (pl. [MASK], [UNK] stb.)

Készítettünk továbbá olyan adathalmazt is, amelyben a különböző helyi értékek számjegyeit különbözőképp jelöltük: az egyes, tízes és százas helyiértékű számok mögé beszúrtunk egy C, B, és A betűt. Így például a fenti kifejezés a következőre módosul:

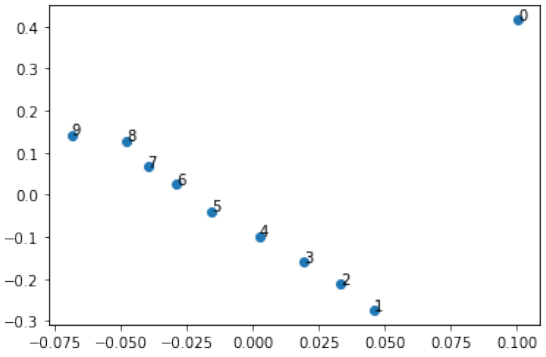
$$((3A8B3C + 2A6B9C)/((1C * 1C) * (6A4B2C - 6A4B1C))) = \\ ((5A7B1C/(3A9B1C/3A9B1C)) + 8B1C)$$

Ezt az adathalmazt nevezzük *abc*-nek. A tokenek itt annyiban változnak, hogy az $0, 1, \ldots, 9$ tíz token helyett $0A, 0B, 0C, 1A, 1B, 1C, \ldots, 9A, 9B, 9C$ harminc tokent használunk.

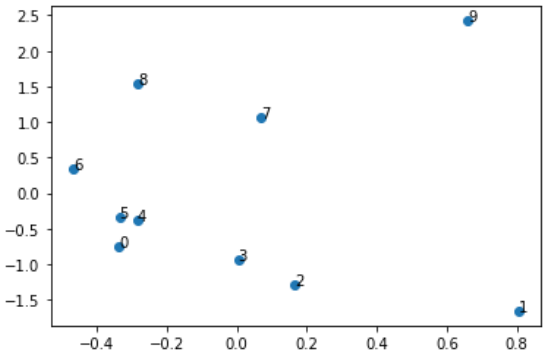
3.2. Eredmények

Számos kísérletet elvégeztünk a kapott beágyazásokon. A tanítás után minden tokenre kaptunk egy-egy vektort. Amikor ezen vektorokat ábrázoljuk, akkor mindig főkomponens analízist hajtunk végre a vektorokon, majd a végeredményként kapott vektor első két koordinátáját ábrázoljuk a síkon.

Egyik legérdekesebb eredmény, hogy a *normal* adathalmazon a számjegyek a BERT esetén nagyjából egy vonalban helyezkednek el a 0 kivételével, mind a statikus, mind a dinamikus beágyazás esetén, ahogy az a 2. ábrán is látható. Hasonló valamelyest megfigyelhető a Word2Vec esetén is, ahol szintén nagyjából egy görbe mentén helyezkednek el a számok, lásd 3. ábra.



2. ábra. BERT statikus beágyazás 0-tól 9-ig

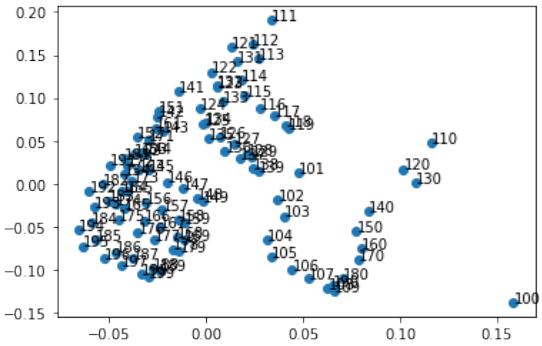


3. ábra. Word2Vec beágyazás 0-tól 9-ig

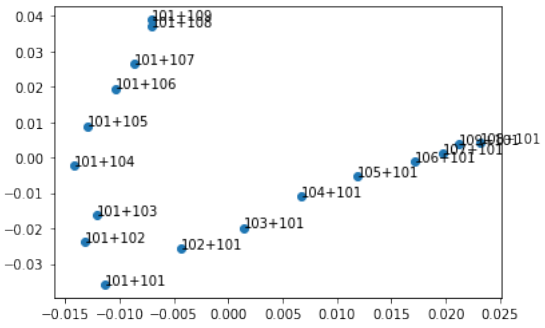
Érdekesség azonban, hogy ugyanez a *relation* adathalmazon nem figyelhető meg.

Amennyiben több számjegyet írunk egymás mellé, akkor az figyelhető meg, hogy a 110 lényegesen közelebb helyezkedik el a 120-hoz, mint a 109-hez, lásd 4. ábra. Továbbá ha a műveleteket is figyelembe vesszük, akkor azt látjuk, hogy a modell nem volt képes a kommutativitás megtanulására, ugyanis a $101 + 109$ és a $109 + 101$ nem egy helyre kerültek, lásd 5. ábra.

Ha az *abc* adathalmazon tanítjuk a modelleket, akkor (az elvárásokkal megegyezve) jól észrevehető, hogy az A, B, C helyiértékek egyértelműen három különböző blokkban helyezkednek el. Felmerülhet a kérdés, hogy ha az XA, XB, XC vektorokat átlagolom ($1 \leq X \leq 9$), akkor vajon ugyanazokat a vektorokat kapom-e, mint *normal* megfelelő vektorai, legalábbis elforgatás erejéig. Azaz létezik-e A mátrix, hogy $M_{abc}A \approx M_{normal}$, ahol M_{abc} a *relation* tokenjein értelmezett vektorokon vett átlagolás utáni mátrixot jelöli, melynek mérete megegyezik az M_{normal} mátrix méretével, amely pedig a *normal* tokenjein értelmezett vektorok mátrixát jelöli.



4. ábra. BERT dinamukis beágyazása 100-tól 199-ig



5. ábra. BERT dinamikus beágyazása: számok összeadása

Az eredmények vegyesek voltak, az összes vektorra egyszerre nem lehetett ilyen A mátrixot találni, de a tokenek egy bizonyos részhalmazára viszont lehetett.

Hivatkozások

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

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

[3] Miroslav Olsák, C. Kaliszyk, and Josef Urban. Property invariant embedding for automated reasoning. In *European Conference on Artificial Intelligence*, 2019.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.