A 2018 őszi egyetemi hallgatóknak és érdeklődő Morgan Stanley kollégáknak szóló NLU kurzus egy innovatív szövegértési modell közös megismerésére, tesztelésére, továbbfejlesztésére irányul. Az elmúlt 2 év során implementált modellből kiindulva remélhetőleg izgalmas közös munkának nézünk elébe.

**A concept háló modell háttere egyéb irányzatokhoz képest**

Mostanában szokatlannak ható megközelítésben modellezzük és implementáljuk a racionális emberi gondolkodást. Szokatlan annyiban, hogy az elmúlt 20 évben inkább az egyes részfeladatokat igen magas szinten megoldó specializált AI megoldásokról lehet hallani. Ezzel szemben igen kevés erőfeszítés történik az „AI szent grál”-jának is nevezhető, a teljes emberi intelligenciát megcélzó kutatásra (lásd még wikipédia: AI-complete). Ennek hátteréről alapos összefoglaló a wikipédián az *AI winter*cikkben.

A concept háló modell közvetlen célja egy implementált NLU (Natural Language Understanding) rendszer, amely az 5 éves gyermek szintjének megfelelő természetes nyelvi outputot képes produkálni angolul.

A szoftver inputja: természetes nyelvű angol szöveg, és modell paraméterek, szabályok, írásban megadva. Outputja: természetes nyelvű angol szöveg, amely az inputban megfogalmazott kérdésekre, felszólításokra adott válasz.

A tesztelés folyamata: Szövegértés teszttel mérjük, hogy a szoftver képes-e az 5 éves gyerek szintjén gondolkodni. Ehhez 5-8 éves szintű szövegeket adunk inputnak, és kérdéseket, amelyekre 5 éves szinten várjuk a választ. A rendszer válaszol természetes nyelven, ezt összehasonlítjuk valódi 5 éves gyerekek teljesítményével.

Az 5 éves szintű teljesítmény mellett a rendszer tervezésénél fontos szempont, hogy felnőtt szintig skálázható, továbbfejleszthető legyen, illetve ilyennek tűnjön (mivel ezt értelemszerűen teljes körűen nem tudjuk még tesztelni).

Van még egy követelmény amit szem előtt tartunk, mégpedig, hogy a modell részleteiben és részfunkcióiban is minél több analógiát, részmodellt tartalmazzon az ember ismert kognitív jelenségeire nézve. Ez egy meglehetős újdonság a megközelítésben, ugyanis közvetlenül felhasználjuk a kísérleti kognitív pszichológia eredményeit, ami egy implementált AI megoldás esetén nem gyakori eljárás.

**Hol tartunk most**

Van egy elég részletesen kidolgozott modellünk, amely a modell kidolgozottsági szintjéig implementálva is van, és a fent leírt tesztet részben teljesíti. De csak kisebb részben, tehát igen sok munka van hátra, és most következik a munka izgalmasabb (nemtriviális) része.

**A concept háló modell röviden**

A modell (és az implementáció) 3 modulja és azok kapcsolódása:

1. Mentalese fordító. Input: angol természetes nyelvű szöveg. Output ugyanaz a szöveg „mentalese” nyelven. (A mentalese a gondolkodás belső nyelve, ahogy Pinker leírja a *The language instinct* c könyvében, ez legálisan letölthető. Lásd továbbá *Language of thought hypothesis* a wikipédián.) Ez nem egy gondolkodó modul, inkább a konkrét nyelv (angol) szintaktikáját dolgozza fel. A szóalakok a mentalese szövegben nagyjából ugyanúgy szerepelnek, mint az angolban.
2. Központi modul. Input: az első modul outputja, mentalese-ben írt szöveg és kérdések. Output: mentalese nyelvű válaszok a kérdésekre. Ez a gondolkodó modul.
3. Visszafordító mentalese-ről angolra. Input a 2. modul outputja, output annak angolra fordított változata.Ez az első modul inverze.

A továbbiakban csak a központi modullal foglalkozom.

**Adatszerkezet**

Egy igen egyszerű reprezentációt alkalmazunk a gondolkodás és a nyelv modellezésére. Reprezentációnkban csupán két elemi adattípus van, az egyik a *szóalak* (az input szöveg szóalakjai a mentalese fordításban is változatlanul megjelennek), a másik a *concept,* amely az összes adattárolási és feldolgozási feladatot ellátja. (Innen kapta a modell a *concept háló* nevet.)

**Szóalak .** A szóalak objektum adattartalma igen egyszerű:

1. A szó, mint karakterlánc. A mentalese fordítóban az input szavak minimális előfeldolgozások esnek át. Ezért tárolnunk tipikusan csak a szótöveket kell. Azonban az angol toldalékok egy része külön szóként jelenik meg.
2. A szó jelentéseire mutató mutatók. Ha egy szónak több jelentése van, akkor több mutató tartozik hozzá.

**Concept, gondolat** (= fogalom, állítás, szójelentés, tudás, koncepció, mondat, bekezdés, történet – ez mind, egyetlen reprezentációval lefedve). A modell alapvető építőköve. Neve a modellben: *concept.* Azért ez a neve, mert a kognitív pszichológiai irodalom concept szavának megfelel. A concept képességei:

1. Egy szó egy jelentése egy concept.
2. Több concept egymáshoz kapcsolódhat, és ezzel új concept jön létre. A modellben minden érdekes információ ezekben a kapcsolatokban (vagyis a különböző szinteken összetett conceptekben) van. Ezzel egyre összetettebb, hierarchikus concept mintázatokat lehet építeni. Ezeken a mintázatokon hajtja végre a modell valamennyi működését.
3. A conceptet mentalese nyelven lehet megadni. Ha ember írja le a conceptet, akkor ez egy mentalese nyelvű string, például C(a,b). A rendszer ezt objektum instance-ként tárolja.
4. A conceptek összekapcsolódásának vannak fajtái. Kb 20 féle kapcsolat van. Például: azonosság, jele D. Ez a mentalese: D(x,y) azt fejezi ki, hogy az x és az y concept azonos (és közben maga D(x,y) is egy concept). Egy másik fontos fajta az osztályba tartozás (C). C(elefánt1, állat1) azt fejezi ki, hogy az elefánt1 concept (ami lehet az elefánt szó egyik jelentése) az állat1 osztályba tartozik. Végül még egy példa: az „A” típusú conceptet a cselekvő és a cselekvés összekapcsolására használjuk: A(he1,run1) a mentalese megfelelője a „*he runs”* mondatnak.
5. A kapcsolat fajtán túl néhány további attribútuma is van a conceptnek. Pl: mennyire igaz, mennyire releváns, mennyire ismert, mennyire mond ellent más concepteknek, mennyire egyedi és mennyire általános. Ezek nem bináris, hanem többfokozatú attribútumok.
6. A concept akármilyen nagyra nőhet. Egy egész könyv is lehet egyetlen concept.
7. A kisebb conceptek, a mondat szintjéig, tipikusan úgy állnak elő, hogy a nyelv szintaktikája eléggé behatárolja (de nem teljesen egyértelműen) hogy mi, milyen kapcsolatban van egymással a mondatban. (pl *An elephant is an animal.*)

**Rövid távú memória.**(working memory.) Jele WM. Az első helyszíne a feldolgozásnak. Ide tároljuk az inputban frissen kapott concepteket (és majd más eredetű concepteket is). Technikailag ez egy concept lista, amiből időnként törlünk. A további feldolgozás a teljes WM tartalmat figyelembe veheti, vagy annak egy részét.

**Hosszú távú memória.**Jele KB. Ez egy másik concept lista, amelyből nem törlünk – ez a hosszú távú emlékezet, a tanulás helye. Ide akkor írunk, mielőtt valamit a WM-ből törlünk. Azt írjuk ide, amit hosszú távra meg akarunk tanulni. A feldolgozásban nem vesz részt a teljes KB tartalom, csak a jelenlegi WM tartalom által aktivált része.

**Szóalakok listája.**A szóalakokat tároljuk egy listában. Ha az inputban megkapjuk a következő szót, kikeressük ebben a listában, és hozzá kapcsolva megtaláljuk az összes jelentését, amik már conceptek. Ezzel indul a feldolgozás.

**A feldolgozás fő mechanizmusai**

A gondolkodás folyamatábrája a következő. Ezeket a lépéseket fogjuk a kurzus során megismerni.

Step 1. Mentalese expression

For all concepts: decrease all pre-activation in WM and KB.

If new paragraph started: erase all activation and pre-activation in KB.

If new input is ambigous (for example a word with multiple meanings, or mapping required): initiate branching of WM to evaluate all options

Step 2.

Store the new input in WM on all living branches (we call this current concept).

Search for the currentconcept in WM and KB. Update probability and other counters. Update consistency counter of current concept.

For words not found, store them in KB and assign a meaning concept.

Generic concepts (reasoning blueprints) found: mark them activated

Specific concepts found: mark them activated

Search for most relevant concepts among children of activated concepts, mark them activated

All children of concepts activated in this round: increase pre-activation level. Those exceeding threshold: mark them activated.

Step 3. Consider next activated generic concept (reasoning blueprint).

If the current concept now fully matches the reasoning pattern, generate implication concept. This becomes new input, go back to step 2.

Consider next activated concept.

If that one matches the reasoning blueprint, go to completness check.

No more reasoning available.

Update the consistency of all branches in WM. For worst branches, discontinue the branch.

Current concept is not a question.

Current concept is a question.

Step 4. Delete a portion of WM if necessary, store a selection of deleted concepts in KB.

Step 5. Search again for the concept pattern defined by the question. If found, deliver the answer.

Back to step 1.

**Aktiválás.** A rövid távú memóriába bekerülő új conceptet megkeressük a KB-ban, és a hozzá kapcsolódó, valamennyire releváns concepteken az aktiválás szintjét emeljük (majd a valamennyire aktivált többi KB-beli concept aktiválás szintjét csökkentjük). Bizonyos küszöb fölé aktivált KB tartalom vesz részt a jelenlegi feldolgozásban. Ha az input kérdés, akkor a WM-ben és a KB-ban szélesebb körű aktiválást hajtunk végre.

**Szó jelentés meghatározása (word sense disambiguation, WSD).** Ez hagyományosan az NLP irodalom kedves területe, főleg mert sok kész tanító és tesztanyag van hozzá. A modellünkben az 1. modul nem végzi el a WSD-t, hanem a szóalakot a mentalese szövegben átadja a 2. modulnak. A 2. modul tud a szó ismert jelentéseiről (valamint időnként észreveszi, hogy egy eddig ismeretlen új jelentésről van szó), és meg kell határoznia, hogy ezek közül melyik használandó most. Itt történik a WSD az alább részletezett bizonytalanság feldolgozás és hipotézis értékelés segítségével.

**Bizonytalanság, többértelműség (ambiguity).** A természetes nyelvben, és a gondolkodásban, nemcsak a szó jelentése bizonytalan és többértelmű, hanem minden más is. Nem egyértelmű a szintaktika, az igazság, és semmi más sem. A bizonytalanság egy részét a conceptek attribútumaival próbáljuk megragadni (például az igazság nem bináris hanem többértékű), ez sokszor inkább fokozatosságot jelent. Másfajta bizonytalanság a szó többértelműség vagy a szintaxis többértelműsége, mert ezeknek legtöbbször egyetlen jó megoldása van (néha nem, ez a humor, amit ezáltal meg tud ragadni a rendszer, bár ez csak melléktermék). Ez a többértelműség az alábbi hipotézis értékeléssel kezelhető.

**Hipotézis értékelés (branching).** A többértelmű szavak példáján mutatom be. A szóalakhoz kapcsolódóan listázzuk az összes értelmét, és a szöveg eddigi feldolgozását annyi hipotézisre ágaztatjuk el, ahány értelme van. Minden ágon folytatjuk a feldolgozást, különösen az alább ismertetett reasoning-et, melynek során a teljes korábbi input, a KB, és a jelenlegi hipotézis alapján következtetéseket vonunk le. Minden következtetést (ezek conceptek) összehasonlítunk a teljes WM-mel és a KB releváns részével arra nézve, hogy mennyire mond ellent nekik (lásd lentebb igazság), és ennek alapján az egyes ágak között különbségeket tudunk megállapítani. Hamarosan a rosszabb ágakat elvetjük, és egy vagy pár jobb ágon folytatjuk csak a feldolgozást. Ez a folyamat produkálja a kognitív pszichológusok által megfigyelt text comprehension (szövegértés) jelenségeket.

**Következtetés (reasoning).**Az a feldolgozás, amelynek során néhány input concept és egy általános conceptként rögzített szabály (reasoning blueprint) alapján új conceptet hozunk létre, és WM-ben tároljuk. Nézzünk egy példát a reasoning működésére: Legyen KB-beli concept A(animal1,live1) (=az állatok élnek). Jöjjön az inputban hogy C(elephant1, animal1) (=az elefánt állat). Van egy olyan általános conceptünk, ami szerint C(%1,%2) és A(%2,%3) fennállásából következik A(%1,%3). A két input az általános szabály mintát kielégíti. Ezért a reasoning eredménye az A(elephant1, live1) concept. A reasoning egy igen sok részletkérdést rejtő bonyolult témakör.

**Válaszadás mentalese nyelven (question answering).** A régi eredetű KB tartalom, a jelenlegi input, és a jelenlegi reasoning alapján, ha a legutolsó input egy kérdés, akkor mentalese-ben válaszolunk. Ennek során megkeressük a választ WM-ben és KB-ben, és rámásoljuk az outputra.

**Háttértudás (world knowledge, common sense).** A hasonló próbálkozások egyik Achilles-sarka. Olyasmiket kell tudjon a rendszer, hogy ha valaki bement egy ajtón, akkor az vagy nyitva volt, vagy előbb kinyitotta. A modellben erre egy első ránézésre fapados megoldás van: az 5 éves szinten szükséges common sense természetes nyelvi inputként tanítandó meg. Vagyis egyrészt le kell írni az 5 éves gyerek által kapott nyelvi inputot. Másrészt sok olyan inputot kell írni, amit egy 5 éves igazi gyerek nem kap, mint nyelvi inputot. (Például: *If a person moves toward a solid wall, he will stop at the wall and he is not able to move forward. A wall can have a large gap which means it is not solid. A large gap is taller and wider than a person. If a wall has a large gap and a person moves toward the gap, he will not stop at the wall but move forward.)* Látható a modell egy fontos limitációja, mely szerint az ember nem-nyelvi inputjait (például a vizuális inputot és a vizuális képzeletet) is mentalese-be kell hogy konvertálja. Ez egyfelől nehézség, másfelől egy létező kognitív jelenség analógiája (mely szerint egy történet-mesélés feladatban csekély a különbség a humán teljesítményben attól függően, hogy a történetet felolvasták, vagy némafilmen vetítették).

A háttértudás tehát nem más, mint alkalmas módon, emberek által angolul írt input a KB-ben tárolva.

**Indukció, analógia, intuíció, asszociáció, kreatív gondolkodás.** A reasoning deduktív jellegű következtetés. Az indukció más. Egyik esete, ha egy állítás egy speciális esetről található az inputban, de az általánosra következtetünk. Pl. Az input szerint az elefántnak 4 lába van, és ebből arra is következtetünk, hogy az állatnak 4 lába van. Vagy egy asszociáció példa:régen tudjuk, hogy egy konkrét vegyipari termék 10 kg-os zsákokban tárolva 10 évig biztosan eltartható, 100 kg-os kiszerelésben viszont pár éven belül néha felrobban. Most megtudjuk, hogy a gabonasilók néha felrobbannak, és kérdés, ez hogyan előzhető meg. Az asszociáció a *felrobban* concept környezetének aktiválásával indul el, és oda jut(hat), hogy a rendszer a válaszának részeként javasolja kipróbálni a gabonasilóban lévő gabona kisebb rekeszekben történő tárolását.

**Igazság.**A rendszer igazságfilozófiája a következő: egyre inkább igaz az a concept, ami egyre többször, és főleg határozottabban igazként állítva fordul elő az egyre megbízhatóbb forrásnak tekinthető inputban. Az eredmény a modellben az, hogy a következő mondat: *All dogs have four legs*  egy megbízható inputból egyetlen előfordulással is egy igen magas, nehezen megingatható igazságú conceptet eredményez (ezzel szemben a *He said all dogs have four legs* már kevésbé, mert ez a fajta forrás legtöbbször csak közepes lehet). Ez teljesen analóg a gyerekek tanulásával.

A megingathatóság abban az értelemben nem nehéz, hogy újabb egyetlen állítás tudja ezt módosítani, ha elég határozott és jó forrásból való. Ezen a ponton az irodalomban lévő modellek kezdenek nehézségbe ütközni, hogyan kezeljék a háromlábú, a lábatlan döglött kutyákat, azt a kutyaképet amire a lábat nem rajzolták rá, és a kivételek további végtelen sorát. Ezt a modellünk megfelelő reasoning-funkciókkal, valamint a lejjebb részletezettáltalánosság (generality) attribútummal kezeli. Továbbá a modellben az igazság nem bináris, hanem fokozatos, ennek is van némi szerepe, de nem ez a legfontosabb.

**Általánosság, egyediség, osztály hierarchia és kivételek kezelése.** A szó jelentéshez tároljuk az általánosságot, ez a conceptek egy attribútuma (g). Így a *dog* szó legszokásosabb *dog1* jelentéséhez g=1 lesz megadva. (Azt, hogy ez a legszokásosabb jelentés, onnan lehet igen könnyen tudni, hogy a legtöbb összetett conceptben szerepel. Más szóval, erről a jelentéséről tudjuk a legtöbbet.) Amit erről tudunk, az régi ismeret, a KB-ben van. Maradjunk ennél a jelentésnél.

* Az egyik fajta kivétel az egyedi kutya: az inputban egy konkrét kutyáról, dögről, képről stb van szó. Vegyük csak azt a problémát ha ennek az inputnak bizonyos információira hosszú távon akarunk emlékezni. Akkor egy Q(a1,dog1) (g=0) conceptből indulunk ki, ami megfelel annak hogy ”*a dog”*, magyarul *egy konkrét kutya*. (Eltekintünk most az „*a”* szó több jelentésétől.) Ezt a KB-be tesszük, nyilván közvetlenül kapcsolódik a dog1-hez. Amit megtudunk bármely egyedi kutyáról, azt a Q(a1,dog1)-hez kapcsoljuk, nem a dog1-hez (kivéve amikor indukálunk). Ezzel minden elég jól megoldódik:
  + a dog1-ről meglévő ismereteink változatlanok.
  + mégis könnyedén aktiválható minden, amit megjegyeztünk egyedi kutyákról.
  + ha kell, a gyakoribb egyedi jellemzők megkülönböztethetők a ritkáktól.
  + összekeveredhet a különböző egyedi kutyákról megjegyzett információ (éppen mint az embernél), de nem feltétlenül keveredik össze. Pl egy háromlábú egyedi kutya infó tárolódhatott így: D(F(Q(a1,dog1),Q(three1,leg1)), I(which1, A(I1, F(saw1, R(in1, Q(the1, garden1)))))) azaz *The dog with three legs which I saw in the garden.*  (legyen ez a concept jelölve 3DL-nek.) Az aktiválás sorrendje: dog1 -> Q(a1,dog1) -> F(Q(a1,dog1),Q(three1,leg1)) vagyis háromlábú kutyák összekeveredve -> 3DL. Ha éppen erről az egyedi kutyáról tudjuk, hogy szereti a spenótot, akkor ezt pontosan így fel fogjuk tudni idézni.

(Implementáció első verziója készen van.)

* A másik fajta kivétel az alosztály-jellegű. A dog1 concept közvetlen kapcsolata egyrészt a F(dog1,Q(four1,leg1)) vagyis *minden kutya négylábú = négylábú kutyák* magas igazsággal. Másrészt ott van a F(dog1,Q(three1,leg1)) vagyis *minden kutya háromlábú = háromlábú kutyák* kicsi igazsággal, de itt bejön egy másik igazságdimenzió, a „létezik”, amely dimenzióban ez utóbbi concept lehet magas. (A témához tartozik még az egymást kizáró conceptek kérdése de ezt most kihagyom.) (Implementáció első verziója készen van.)

**Mapping.**(coreference resolution) A szövegértés egyik alapvető jelensége (embernél természetesen), hogy a szöveg szavait a korábbi szavakkal „azonosnak” vesszük. Például: *Joe is three years old. He loves dogs.* Itt egy gyakori eset, hogy a *he* szót úgy értjük, hogy az is *Joe*-t jelenti. Ez a mapping. Ez adja az egyik igen fontos mondatok közötti kohéziót, és fokmérője a szöveg megértettségének.

A modell a mapping-et hipotézis értékeléssel kezeli. Felismerjük a potenciálisan (inkább vagy kevésbé) mappingre szoruló szót, az input valamely környezetében kiválasztjuk a lehetséges párokat hozzá, és mindegyikkel indítunk egy hipotézis ágat. A rossz ágakat hamarosan elvetjük. (Implementáció első verziója készen van.)

**Mese, megszemélyesítés, átvitt értelem, metaforikus szóhasználat.** Mindenképpen kell legyen megoldásunk többek között arra, hogy egy mesekönyvben a teknősbéka beszél. Sőt, amit mond és csinál, arról emberre kell tudni indukálni egyes esetekben, más esetekben m eg el kell k erülni ezt az indukciót. Ez nincsen elegánsan kidolgozva, egyelőre jobb nem jutott eszembe mint a kivétel-kezelési módszerek (alosztály, egyedek). (Implementációja nincsen.)

**Mondatközi és magasabb kapcsolatok.** Mondat szintje felett nehezedik a conceptek létrehozása, az 1. modul egyre kevésbé alkalmas erre.

* Még az 1. modul tud kezelni olyanokat mint például ok-okozat: *He ran out of water three days ago. So he was happily surprised to see the lake.* A „so” miatt a két mondat között ok-okozat típusú concept létrehozható.
* Ha a fenti példában nincs „so”, akkor a 2. modul tudja az ok-okozatot felállítani, de csak meglévő KB tudás alapján.
* A mapping sokszor összekapcsol egész mondatokat.
* Egyes conceptek kifejezetten azt jelentik, hogy magas szinten kapcsolatot kell létrehozni. Ez egyfajta önhivatkozás, nincs a modellben kidolgozva. Például egy bekezdés azzal kezdődik, hogy *In the previous paragraph we only focused on .....*

(Implementációja nincsen.)

**Az adatbázis méretéről, és a conceptek bonyolultságáról.**

A probléma: Nem lesz-e egy idő után hatalmas a KB? Nagy KB mellett nem válik-e hatalmassá a WM egy szöveg feldolgozása során, különösen ha minden lehetséges reasoninget elvégzünk? Ha egy könyv egyetlen concept, nem lesz-e túl részletes a feldolgozás, ha feltesznek valami egyszerű kérdést a könyvről, mint például hogy miről szól?

A KB-be kétféleképpen keletkezik adat: 1. benne volt az inputban (vagyis olvastuk) 2. reasoning útján. Az olvasott KB tartalom a természetes nyelvi szöveg méretével kb lineárisan nő csupán. Egy 2 szavas mondat kb 3 concept ("I go." ebből az I, a go, és az A(I,go) concept keletkezik.) Egy 6 szavas mondat kb 11 concept ("The boy went home after six." 6 concept a 6 szó, ezen felül Q(the,boy), F(go, home, T(past)), T(after,six), F(F(go,home,T(past)), T(after,six)), és végül A(Q(the,boy), F(F(go,home,T(past)), T(after,six)).    ) Ugyan a conceptek így papíron leírva egyre hosszabbak (mert egyre mélyebben összetettek), de az ábrázolásuk semmivel nem igényel nagyobb helyet mint a legegyszerűbb conceptek. Ez azért van, mert papíron a szülő concepteket újra és újra kiírom, de az implementációban csak hivatkozok rájuk. A mondatok közötti és magasabb conceptek hozzájárulása kicsi (egy mondat többnyire 1 másik mondattal képez 1 db conceptet).

Így tehát egy könyv olvasott conceptjei csak annyi helyet foglalnak mint konstans \* a könyv hossza angolul. De itt vannak a reasoning útján képzett conceptek. Ezek maximális száma legalább négyzetesen arányos a szavak számával. Már ez is túl sok. Elméletileg nincsen szükség a resoninggel keletkező conceptek KB-be tárolására mert új információ nincs bennük. GYakorlatilag pedig a következő a tervem (ez még nincs implementálva): minden conceptnek van egy "relevance" (lényegesség) tulajdonsága, röviden r. Képzeljük mondjuk 5 értékűnek 0-tól 4-ig.  Csak azokat a resoning útján keletkező concepteket tároljuk KB-be, amelyeknek legalább egyik szülőjére r elér egy küszöbértéket (pl 4). Ezután a modellben hangolható, hogy mennyi reasonelt conceptet tárolunk. Erre nemcsak a küszöb fog hatni, hanem az r érték kiszámítási módja. Ennek a lényege, hogy egy concept a többihez képest felette lényeges, ha a gyerekeinek száma eléri egy küszöbszörösét az átlagos (medián) gyerekszámnak. Példa ami érzékelteti ennek működését: megfelelő alapképzés után a rendszer elolvassa a Háború és Békét. A teljes input gyakori szavainak (pl: ember) több ezer gyereke lesz, a ritka szavaknak (pl: bekerítés) kb 1 vagy kb 10. A medián gyerekszám 20. A küszöb pl 5, igy a releváns (r=4) concepteknek min 100 gyereke van. Ez egy kicsi hányada lesz az összesnek.

Igy végül se 1 se 2 hozzájárulása nincs még négyzetes se.

A másik érdekes kérdésed, hogyan egyszerűsítünk le egy olyan conceptet, mint a "Háború és béke" (H&B). Valóban le kell egyszerűsíteni, hogy elég jó analógiában legyünk az emberi gondolkodással, és hogy ha akarjuk, ne emlékezzünk az egész könyvre. Ezt is a relevanciával tervezem (nincs még implementálva). Vegyük a H&B egy fejezetét, ebben mondjuk Katyusa férjhez megy. Ez a fejezet így lesz egyetlen rövid conceptre egyszerűsítve: 1. elolvassuk az egész fejezetet és teljesen reprezentáljuk. 2. Vesszük a fejezet top 10 legjellemzőbb conceptjét: ezek azok, amik  a fejezetben relevánsak, de az egész nyelvben nem. Ilyenek lesznek a Katyusa, és a férjhez menés. A férjhez menés biztosan releváns lesz a fejezetben akkor is ha csak 5-ször van szó szerint kiírva, mert rengeteg mondat stb kapcsolódik a férjhez menéshez. Végül vesszük ezen top 10 concept legjobb közös gyerekét (a 10 conceptből sok legyen benne, más conceptekből kevés), ez lesz a leegyszerűsített fejezet. Egyetlen mondatban kimondva pl ilyesmit kaphatunk, hogy "Katyusa Moszkvában megismerkedik Jevgenyij fiával, akihez férjhez megy."  Ez lesz a fejezet egyszerűsített változata. Egyszerre megjegyezhetjük ezt is, és a teljes kifejtést is. Ha kérdés jön, akkor az egyszerűsített conceptek a H&B kontextusában relevánsabbak, mint a teljes kifejtések, ezért inkább azok alapján válaszolunk.

Az aktiválás mechanizmusa nem más, mint egy relevancia küszöb feletti kontextek aktiválttá jelölése. Még tartozik hozzá egy bejárási sorrend. Az aktiválás a fenti két kérdéssel abszolút összefügg. Az aktivált conceptek alapján végezzük a reasoninget (és emiatt a kérdésre válaszolást is).

**Tanító anyag.** Az 5 éves szint elérését valahogy úgy képzelem, ahogy egy gyereket oktatnak az ovodában és az első osztályban a világ dolgaira, Csak éppen sokkal több téma van. Tehát fel kell sorolni a témaköröket (amiknek persze nincs világos határa), és minden témakörre kell írni egy oktató anyagot (jó esetben pár oldalt), a közben felmerülő common sense hiányt folyamatosan pótolva. Például egy témakör lista eleje, ha az első témakör a *reggeli: ebéd, uzsonna, vacsora, evés, ivás, tevékenységek reggel, fogmosás, ébredés, öltözködés, alvás, ágy, pizsama, ruha, piszok, tisztaság, mosakodás, szék, asztal, tányér, evőeszköz, ételek, italok, konyha, étkező, szoba, hálószoba, lakás, ház, kint, bent, udvar, utca, óvoda, anyu, apu, testvér, család, családtagok, gyerekek, óvó néni, ismerősök, idegenek, idegen gyerekek, emberek, állatok, kutya, macska, ........................*

Az 5 éves szinthez ezeket a szövegeket meg kell írni, készen nem lesznek igazán jó szövegek megtalálhatók.

**Mi a modell buktatója?**

Úgy látom, a fő várható buktató a skálázhatatlanság, ennek csomó esete lehetséges:

* common sense betáplálásához nem várt mennyiségű input kellene
* a rendszer béna viselkedésének javításához rengeteg tesztelés, programozás kellene
* a KB növekedésével kellemetlenül nő a feldolgozási idő az aktiválás ellenére is, illetve kisebb körű aktiválás jelentősen rosszabb eredményre vezet
* az input növekedésével meglepő mértékben nő a KLB mérete
* ha a fentiek mind megoldhatók, akkor még mindig lehet hogy az 5 éves szintű outputhoz igen sokáig tart megírni a tanító anyagot.

**A concept fajták listája**

A rendszer finomítása során bármikor be lehet vezetni új concept fajtát. Ekkor a hozzá tartozó reasoning szabályokat kell megírni, és, amennyiben az 1. modul (mentalese-re fordító) működik, abban is fel kell venni azokat a fordítási szabályokat, amelyek erre az új fajtára fordítanak.

A jelenlegi concept fajták:

1 – W : word, szó jelentése

2 – S : similar, két concept hasonló

3 – D : identical, két concept azonos

4 – C : class, egy concept magasabb hierarchia szinten lévő osztály, aminek alosztálya vagy egyedi példánya a másik concept.

5 – F : feature, egy conceptnek van egy tulajdonsága.

6 – Q : quantifier, egy conceptből van valamennyi, vagy egy egyedi conceptet specifikál. Például Q(three, people) három ember, Q(this, person) ez az ember, Q(my,wife) az én feleségem, Q(the, fox) pedig attól függően hogy a *the* melyik jelentéséről van szó, lehet egy konkrét róka, vagy lehet a róka mint állatfaj.

7 – A : action, egy concept cselekszik valamit, vagy egy concept időbeli változás folyamatában vesz részt.

8 – I : impact, egy concept valami cselekvésnek, vagy időbeli változás folyamatnak a tárgya (ami megváltozik).

9 – R : relate, egy módosító a concepthez képest kijelöl egy viszonyt, például térbeli irányt. Pl: R(on, Q(the,table)) = az asztalon. Példa nem térbeli viszonyra: R(with, me) = velem.

10 – T : time, egy concept időbeli tulajdonságot jelent, például T(now). Ezt módosíthatja egy módosító, például T(before, yesterday).

11 – P : part, egy conceptnek van egy része, vagy birtoka.

12 – M : more, egy concept egy mérték szerint kisebb/nagyobb viszonyban van egy másikkal. Pl M(count, Q(my, money), Q(his, money)) = nekem több pénzem van, mint neki.

13 – IM : implication, egy concept az ok, egy másik concept a következmény, okozat. Elégséges feltétel.

14 – N : necessary, egy concept szükséges (de nem feltétlenül elgéséges) feltétele egy következménynek.

15 – V: relevance, ez egy elvont viszony arra az esetre, ha más módon a kontextusban való értelmezés lehetősége nem eléggé kifejezett. Ekkor egy concept speciális kontextust jelent, amiben a másik conceptnek különös jelentősége lehet.

16 – AND : and, és viszony.

17 – NOT : not, tagadás.

18 – OR : or, vagy viszony.

19 – XOR : exclusive or, kizáró vagy viszony. Csak arra használjuk, ha conceptek egymást többé-kevésbé kizárják. Pl F(person,XOR(infant,adult)) azt jelenti, hogy egy ember általában vagy gyerek, vagy felnőtt, de nem mindkettő.

Az AND, OR, NOT viszonyokat nem formális logikai értelemben, hanem az átlagos humán használatnak megfelelően kell használni. A kettő sokszor átfed, de például formálisan F(AND(x,y),y) –ből mindn esetben következik F(x,y), míg a rendszer nem kell ehhez mereven ragaszkodjon, lehet számtalan kivétel amikor ez nem egészen teljesül.

**A szoftver ismertetése (a 2. fő modul)**

Az eddigi kódunk itt található: <https://github.com/NatLanGroup/NatLan>. A nyelv python, elvileg 2.7 és 3 verzió alatt is fut. A 2. főmodulhoz semmilyen külső library nem kell. Az én email címem foris64@gmail.com.

**Jelenlegi képességek**

A szoftver most (2018 augusztus) ezeket tudja, röviden:

**Python modulok, input, output**

natlan.py fő modul, egy ciklust tartalmaz, ami az input file sorain megy végig.

gl.py a minden más modulban szükséges változók definiálásának helye

testing.py az input file előfeldolgozása, az output előállítás utolsó lépése

wrd.py a WL szó lista kezelése.

conc.py a mentalese nyelvű string objektummá alakítása

a conceptek kezelése

a WM és a KB kezelése

a kérdésre válaszolás

reason.py a reasoning funkciók

branch.py a hipotézis kezelő, elágaztató és elágazás megszüntető funkciók

input: (lásd testfile.txt): egy txt file, amelynek egy sora tartalmazhat

* egy angol szöveget e/ után (de ezt nem dolgozza fel a szoftver)
* egy mentalese szöveget m/ után, ez az igazi input
* egy elvárt, jó választ a/ után
* egy kommentet // után

az m/ tekinthető kötelezőnek, a többi opcionális. Egy sorban csak 1 db összetett concept lehet (vagyis 1 mondat). Ez lehet egy kérdés is. A rendszer sorban feldogozza a sorokat, minden sor után csinál reasoning-et és esetéleg elágazásokat, és ha a sor kérdés volt, megpróbál válaszolni a kérdésre. Végül a kérdésre adott választ összehasonlítja az a/ után álló jó válasszal, és az outputban jelzi a jó és rossz válaszokat.

output: megismétli az inputot, hozzátéve:

* a rendszer által adott válaszokat (ha a sor kérdés)
* ezen válaszok viszonyát a jó válaszhoz: OK(jó a rendszer válasza), OK MORE (a rendszer eltalálta a jó választ, de még több válasza is van), MISS (a rendszernek nincs válasza), BADP (a rendszer válaszának igazságértéke rossz).

Futtatás a fő teszt file-lal (testfile.txt): %Run natlan.py testfile.txt

Output: testfile\_result.txt és egy log file (logfile.txt)

**Ki kell próbálni a testfile.txt futtatásával. Egy csomó kérdésre már jól válaszol a rendszer, amit az OK és az OK MORE sorok jelentenek. A tanító és kérdés sorok szándékosan úgy vannak elhelyezve, hogy a testfile.txt elején az van ami már működik, és ahol a hibák elkezdenek sokasodni, onnan kezdve nem működik még.**

**testfile.txt**

* Az input elején reasoning szabályok vannak. Ezek is mentalese-ben leírt conceptek. Ezzel egyrészt a meglévő adatszerkezet a szabályok tárolására is felhasználtuk. Másrészt ezek a szabályok tulajdonképpen ugyanúgy elmondhatók angolul, csak furcsán hangzanak.

Például: először is az IM concept az ok-okozatot jelöli, vagyis IM(x,y) esetén x az ok, y a következmény. Minden szabály IM-mel van leírva.

Például vegyük ezt a szabályt:

IM(C(%1,F(%2,%3))p=p0,C(%1,%2))p=pide1

részleteiben:

* C(%1,%2) azt jelenti, hogy a %2 osztály egy eleme a %1.
* C(%1,F(%2,%3)) azt jelenti, hogy a %3 tulajdonságú %2 osztály egy eleme a %1.

Vagyis az egész szabály azt mondja, hogy: „*Ha egy adott tulajdonságú osztálynak valami az eleme, akkor az a valami eleme az osztálynak a tulajdonságtól függetlenül is.”* Például, ha az elefánt nagy állat, akkor az is igaz, hogy az elefánt állat.

A p=im stb jelölések arra vannak, hogy a feltételek igazságértékeit egy konverziós táblázattal (pclass) fel lehessen használni a következmény igazságértékének kiszámolására.

* Van egy reasoning szabály, ami a szabályok között nincs felsorolva, mert a rendszerbe be van drótozva. Ez a legalapvetőbb szabály, mely szerint *Ha A tagja a B osztálynak, és a B osztályra igaz C, akkor A-ra is igaz C.* Ezt a szabályt olyan sokszor és sokféleképpen kell alkalmazni, hogy külön kód van rá.
* Az inputban utána a tanító mondatok és a kérdések jönnek. Minden kérdésre csakis pontosan a felette lévő tanító mondatokat használja a válaszhoz. A kérdésekből nem tanul, vagyis nem használja fel válaszhoz (ez alól kivétel a „lehet” válasz, ami nem jelent semmit, egyenértékű a „nem tudom”-mal).
* Nincsen kiinduló KB tartalom. Minden tudás innen az input fileból töltődik fel.

**natlan.py**

létrehozza ezeket az instancokat (az egész futás alatt élnek):

* gl.args a globálisan szükséges változók, táblázatok.
* gl.WM a working memory.
* gl.KB a hosszú távú memória.
* gl.WL a szavak listája.
* gl.log a log file.
* gl.reasoning a resoning objektum
* gl.test a teszt file feldolgozás objektum.

futtaja a teszt file-t soronként feldolgozó ciklust:

* megjegyzi az utolsó bekezdés határt a gl.WM.paragraph listában, ez azért kell mert eddig megy vissza mapping során az inputban.
* a mentalese nyelvű stringet objektummá alakíttatja és a WM-be tárolja: branch\_read\_concept
* bizonyos inputot rögtön WM-ből átrakat a KB-ba: move\_relevant
* választ adat a kérdésre: answer\_question
* kiíratja az output következő sorát: write\_result
* végrehajtatja a reasoning és az elágaztatás (branching) funkciót

Ha a sor kérdés volt, akkor a reasoning, branching elmarad.

Ha nem, akkor a kérdésre válaszolás átugrásra kerül, és van

reasoning, branching.

debug kiirásokat csinál a képernyőre. Lásd pl ezt a ciklust:

for wmi in gl.WM.cp

ez a WM-en megy végig, és minden concept fő jellemzőit kiírja.

**gl.py**

default értékek

igazságérték és más attribútumok értéktartománya

p igazságérték tartománya most 0,1,2,3,4

a középső érték a 2, jelentése „nem tudom” = „lehet”

4 a biztosan igaz, 0 a biztosan nem igaz

a concept fajták betűjelének felsorolása és számkódhoz rendelése

reasoning-hez szükséges táblázatok (ezek a következtetés feltételeinek igazság-értékéből határozzák meg a kikövetkeztetett concept igazságértékét)

**testing.py**

a gl.test instance egy pár listát tartalmaz, amelynek j-ik eleme a j-ik input sor valamely adata. Például a gl.test.mentalese[j] a j-ik sorban lévő mentalese concept (mint string).

**wrd.py**

A Wlist osztályban (amből a WL instance készül) az add\_word függvény kapja a mentalese stringben szereplő szót, és a szólistába is beteszi, készít neki 1 db jelentést, és azt a KB-be beteszi, a kettőt összeköti.

**conc.py**

Ebben sokminden van. Ezekből pár dolog:

* Concept osztály. Ennek instancai a conceptek, ezekből tehát sok van és többnyire az egészt futás során megmaradnak.
  + .parent : a szülő conceptek listája. Ez a concept úgy jött létre, hogy a szülő conceptek a .relation relációba kerültek egymással.
  + .child : a gyerek conceptek listája. Valamely gyerek concept úgy jött létre, hogy ez a concept önmagában, vagy pár más concepttel, valamilyen relációba került. Egy conceptnek 1-2 szülője szokott lenni, és pár, sok, vagy rengeteg gyereke. Az idő telésével a szülők nem változnak, a gyerekek igen.
  + .previous és .next : láncok alkotására jó abból a célból, hogy tudjuk követni az elágazásokat, ha vannak. Ha nincs elágazás akkor ezekben a mezőkben nincs új információ. Ha egy conceptet követően elágazás van, akkor a next tartalma többelemű (az össze ág első eleme fel van sorolva).
  + Mivel azonosítódik egy concept? Lsd Kbase osztály leírása.
* Kbase osztály. Ennek postosan 2 instanceát használjuk, a WM-et és a KB-t. A fő mezője a .cp, ez a conceptek listája.
  + KB-ból nem törlünk
  + WM-ből is csak úgy törlünk, hogy a legfrisebb conceptektől kezdve visszafelé
  + ezért ha egyszer egy concept bekerült akár KB-be akár WM-be, a .cp listában az indexe nem változik
  + ezért ezt az indexet használjuk a concept azonosítására, így tehát ezt írjuk például a .parent és a .child mezőbe
  + értelemszerűen a WM és a KB nem keveredhet, egy WM-beli concept szülői és gyerekei is WM-ben vannak
  + ezért amikor WM-ből KB-be másolunk, a parent és child értékeket újra ki kell számolni.
* mentalese string objektummá alakítása és WM-be tárolása:
  + natlan.py hívja a branch\_read\_concept függvényt az aktuális sorban lévő mentalese stringgel
  + branch\_read\_concept minden elágazáson (branch) meghívja a read\_conceptet a mentalese stringgel
  + read\_concept önmagát hívja addig amig szóhoz ér a mentalese stringben (azt tárolja) vagy amíg egy concept össze szülőjét már tárolta, és akkor azt tárolja a WM-be. Feldolgozza ha a mentalese string explicit p vagy r vagy c vagy g értéket adott meg.
* reasoning rule másolása KB-be:
  + natlan.py hívja a move\_rule függvényt a mentalese stringgel
  + move\_rule hívja a copyto\_kb függvényt ami a WM alapján KB-be írja az utolsó reasoning rule conceptet
  + move\_rule hívja a remove\_concept függvényt ami törli a WM-ből az utolsó reasoning rule conceptet
* kérdés megválaszolása
  + natlan.py hívja az answer\_question függvényt az utolsó input sorból a WM-be írt conceptek kezdő és végpozíciójával
  + answer\_question hívja a search\_onbranch függvényt, amely hipotézis áganként külön keresi meg az összes választ. Egy válasz az a concept amely megfelel a kérdés-concept szerkezetének.
  + Többnyire töröljük a kérdést a WM-ből (eddig ugyanis benne volt). Ez alól kivétel az igen/nem kérdés, az benne marad, de nem sok vizet zavar.

**reason.py**

* a reasoning class néhány fontosabb mezője:
  + reason\_processed: hol tart a reasoning
  + recordrel, recordparents: a most előállított új concept adatai
* a reasoning funkció további adatai a conceptekben vannak.
* A reasoning folyamat nagy vonalakban:
  + natlan.py hívja a createConceptRules függvényt. Ez megállapítja, hogy van-e olyan rule (szabály) a KB-ben, amelynek egyik feltétele lehet ez a concept. Ha van, akkor ennek a conceptnek a kb\_rules listájába jegyzi fel ezen szabályokat.
  + natlan.py hívja a perform\_Reason függvényt. Ez:
  + az utolsó meghívása óta a WM-be került minden conceptre egyenként:
  + a jelenlegi concept legyen „new”
  + megvizsgáltatja kell-e elágazást beszúrni, ha kell beszúratja.
  + megvizsgáltatja hogy a „new” concept önmagában valamely rule teljes feltételét teljesíti-e, ha igen, a következményt kiszámittatja és WM-be iratja.
  + végigmegy a WM teljes korábbi tartalmán, az újabbtól a régebbi felé. az ebben a ciklusban aktuális concept legyen „old”. Meghívja az enter\_Rulematch függvényt, amely:
    - ellenőrzi, hogy az old-hoz bejegyzett szabályok mindegyikére, a „new” concept teljesíti-e a feltétel valamely darabját, mivelhogy egy szabálynak akárhány feltétele lehet. Ha igen, az old-hoz ezt bejegyzi.
    - ellenőrzi, hogy most az „old”-hoz van-e olyan szabály bejegyezve, aminek az összes feltétele teljesült.
    - Ha van, a következtetést kiszámítja és bejegyezteti a WM-be.