# Predlog izvedbe gradnje množic in modelov za odgovarjanje na vprašanja v slovenskem jeziku (delovna verzija)

## Zahteve

1. Izdelava oz. dopolnitev korpusov za QA po vzoru SuperGLUE množic
   1. Množice SuperGLUE so bile prevedene, mi bi jih dopolnili iz podatkov projekta Slovenščina na dlani (SND)
   2. Uporaba uveljavljene strukture (enako kot v SuperGLUE)
2. Izdelava modelov za QA in evalvacija le-teh s SuperGLUE
   1. Uporaba arhitekture Transformer (trenutno je najboljši model DeBERTa) in ostalih modernih arhitektur na slovenskem jeziku
   2. Knjižnica Huggingface vsebuje implementacije teh modelov v programskem jeziku Python in omogoča zagon znotraj ogrodij Tensorflow in PyTorch.

## Predlog izvedbe

Najprej bi izdelali oz. dopolnili korpuse za QA po vzoru SuperGLUE množic s podatki iz SND. Konkretno bi se omejili na SuperGLUE množice BoolQ, COPA, MultiRC in ReCoRD. Množica COPA je prevedena v celoti in se že uporablja z modeli na slovenskem jeziku. Množica ReCoRD je najbolj problematična, saj gre za strojno generirano množico, ki pa ni preveč kvalitetna. To množico bi poskusili dopolniti oz. v celoti izdelati iz podatkov SND. Na FRI so ugotovili, da so prevodi množic BoolQ in MultiRC dovolj kvalitetni, vendar najverjetneje zmanjka zadostno število primerov za uspešno strojno učenje modelov. Ti dve množici bi poskusili dopolniti s podatki SND. Trenutno je stanje teh množic in njihovih slovenskih prevodov sledeče:

1. BoolQ
   1. train (angleških=9427, prevedenih=92 => 0,98%)
   2. val (angleških=3270, prevedenih=18 => 0,55%)
   3. test (angleških=3245, prevedenih=30 => 0,92%)
2. COPA
   1. train (angleških=400, prevedenih=400 => 100%)
   2. val (angleških=100, prevedenih=100 => 100%)
   3. test (angleških=500, prevedenih=500 => 100%)
3. MultiRC
   1. train (angleških=5100, prevedenih=15 => 0,29%)
   2. val (angleških=953, prevedenih=3 => 0,31%)
   3. test (angleških=1800, prevedenih=30 => 1,67%)
4. ReCoRD
   1. train (angleških=101000, prevedenih=60 => 0,06%)
   2. val (angleških=10000, prevedenih=6 => 0,06%)
   3. test (angleških=10000, prevedenih=30 => 0,3%)

Načrt glede množic podatkov bi bil torej sledeč:

1. COPA – dopolnitev s podatki SND (najnižja prioriteta)
2. BoolQ – dopolnitev s podatki SND
3. MultiRC – dopolnitev s podatki SND
4. ReCoRD – izgradnja množice (zaradi poenotenega formata, se lahko takšna množica kasneje tudi združi s prevedeno množico ReCoRD)

### Struktura SuperGLUE množic

Ker bi sledili uveljavljeni strukturi SuperGLUE množic, so spodaj navedeni kratki opisi množic in njihovih struktur. Vsaka množica ima tri mape, kjer so shranjeni primeri. Mapa test vsebuje testne vzorce, mapa train vsebuje učne vzorce, mapa val pa validacijske vzorce. Struktura testnih, učnih in validacijskih vzorcev se ponekod razlikuje, zato je v nadaljevanju za vsako množico opisana tudi struktura le-teh posebej.

**BoolQ** – Množica vsebuje krajši odsek besedila s kontekstom in vprašanje, ki se odgovori z »da« ali »ne«. Struktura vsebuje naslednje lastnosti:

* idx (int32)
* label (int64; odgovor, ki je lahko 0=ne ali 1=da)
* passage (string; odsek besedila s kontekstom)
* question (string; vprašanje)

Uporabljena struktura po razdelitvi test/train/val:

* test: (idx, passage, question)
* train: (idx, label, passage, question)
* val: (idx, label, passage, question)

**COPA** – Množica vsebuje krajši odsek besedila ali stavek s kontekstom in vprašanje, ki lahko ima več odgovorov (konkretno sta tukaj 2). Struktura vsebuje naslednje lastnosti:

* idx (int32)
* label (int64; odgovor, ki je lahko 0=choice1 ali 1=choice2)
* premise (string; odsek besedila s kontekstom)
* question (string; vprašanje)
* choice1 (string; prvi možen odgovor)
* choice2 (string; drugi možen odgovor)

Uporabljena struktura po razdelitvi test/train/val:

* test: (idx, premise, question, choice1, choice2)
* train: (idx, label, premise, question, choice1, choice2)
* val: (idx, label, premise, question, choice1, choice2)

**MultiRC** – Množica vsebuje krajši odsek besedila s kontekstom in vprašanje, ki lahko ima odgovor »da« ali »ne«. Tukaj je možnih tudi več odgovorov. Struktura vsebuje naslednje lastnosti:

* idx (dictionary)
  + answer (idx32; indeks odgovora)
  + paragraph (idx32; indeks odseka besedila s kontekstom)
  + question (idx32; indeks vprašanja)
* label (int64; odgovor, ki je lahko 0=ne ali 1=da)
* paragraph (string; odsek besedila s kontekstom)
* question (string; odsek besedila s kontekstom)

Uporabljena struktura po razdelitvi test/train/val:

* test (idx, paragraph, question)
* train (idx, label, paragraph, question)
* val (idx, label, paragraph, question)

**ReCoRD –** Množica vsebuje krajši odsek besedila s kontekstom in vprašanje, ki lahko ima več odgovorov. Zraven so podane tudi entitete, ki vplivajo na odgovor. Struktura vsebuje naslednje lastnosti:

* idx (dictionary)
  + passage (int32; indeks odseka besedila s kontekstom)
  + query (int32, indeks vprašanja)
* passage (string; odsek besedila s kontekstom)
* entities (string; entitete, ki nastopajo v besedilu)
* query (string; vprašanje oz. poizvedba)
* answers (string; možni odgovori)

Uporabljena struktura po razdelitvi test/train/val:

* test (idx, passage, entitites, query)
* train (idx, passage, entities, query, answers)
* val (idx, passage, entities, query, answers)

### Izdelava modelov za odgovarjanje na vprašanja

Modele za odgovarjanje na vprašanja bi izdelali s knjižnico Huggingface, ki vsebuje implementacije vseh state-of-the-art arhitektur. Implementacija bi tako bila realizirana v programskem jeziku Python in z ogrodjem za izdelavo globokih nevronskih mrež Tensorflow ali PyTorch. Oba sta podprta v knjižnici Huggingface in oba omogočata učenje na grafičnih procesnih enotah. Kar se tiče modelov, ki bi jih lahko preizkusili predlagam omejitev na družino Transformer modelov, saj le-ti drastično prekašajo nekoč najboljše LSTM in GRU modele. Konkretno so to modeli DeBERTa(-Ensemble in -1.5B), T5-11B in GPT-3 175B (few-shot). Dodatno bi lahko preizkusili tudi modela MUPPET Roberta Large in MUPPET Roberta Base.