Tokenizer

Հայկ Կարապետյան

Ունենք հետևյալ խնդիրը: Մուտքում ստանալով անգլերեն նախադասությունը, պետք է թարգմանենք հայերեն: Հարց է առաջանում, թե ինչպես կարող ենք ցանցին փոխանցել նախադասությունը: Ցանցը մուտքում ընդունում է թիվ, իսկ մեր նախադասությունը string-է: Այդ պատճառով մեզ անհրաժեշտ է մեր բառերը վերածել թվերի, հետո նաև վեկտորների: Սկսենք նրանից, թե ինչպես ենք categorical տվյալները դարձնում թվային: Ունենք ավտոմեքենային մոդելի անվանումը, շարժիչի ծավալը, գույնը, տարեթիվը, վազքը և պետք է գուշակենք մեքենայի գինը: Սա regression խնդրի օրինակ է: Ավտոմեքենայի բոլոր պարամետրերը՝ բացառությամբ գույնի և մոդելի, թվային արժեքներ են: Մեր տվյալները սահմանափակ են, այսինքն գույներն էլ են սահմանափակ: Կարող ենք կազմել գույների զանգված.

$$\text{գույներ} = [\text{”կարմիր”}, \text{”սև”}, \text{”մոխրագույն”}, \text{”սպիտակ”}]$$

Օգտվելով այս զանգվածից կարող ենք ամեն գույնի համապատասխանեցնել մի թիվ՝ ”կարմիր” - 0, ”սև” - 1, ”մոխրագույն” - 2, ”սպիտակ” - 3: ”սպիտակ” գույնի ավելի մեծ լինելը կարող է ցանցի կողմից ընկալվել, որպես առավելություն, իսկ ”սև”-ի փոքր լինելը թերություն: Այդպիսի մեթոդ կարող էր կիրառվել ordinal տվյալների դեպքում (”bad”, ”normal”, ”good”): Այդ պատճառով ստացված թվերը վերածում ենք one-hot վեկտորների.

$$\text{”կարմիր”} - [1, 0, 0, 0]$$
$$\text{”սև”} - [0, 1, 0, 0]$$
$$\text{”մոխրագույն”} - [0, 0, 1, 0]$$
$$\text{”սպիտակ”} - [0, 0, 0, 1]$$

Այսինքն ունենք գույների բառարան և դրանից օգտվելով ամեն գույնի համապատասխանեցնում ենք one-hot վեկտոր: Նույ կերպ ավտոմեքենաների մոդելների բառարան ենք կառուցելու և ամեն մոդելի համապատասխանեցնելու one-hot վեկտոր:
Վերադառնանք նախադասության բառերը ցանցին փոխանցելու խնդրին: Այս դեպքում մեզ նույնպես անհրաժեշտ է կազմել բառարան: Բառարան կազմելու համար վերցնում ենք մեծ քանակությամբ տեքստային տվյալներ (օրինակ՝ մեկ միլիոն նախադասություն): Առաջին տարբերակն է ամեն տառի համապատասխանեցնել մի one-hot վեկտոր: Բայց այդ դեպքում կառաջանա երկու խնդիր: Առաջինը մեր մուտքային տվյալները մեծ կլինեն: Միջինում մի բառը բաղկացած է 5 տառից և 20 բառից բաղկացած նախադասության դեպքում, առանց բացատները հաշվելու կունենանք 100 մուտքային տվյալ: Երկրորդ խնդիրն այն է, որ այդ դեպքում կորչում է բառի կառուցվածքը: Այսինքն ցանցը ուսուցման ընթացքում կարող էր հասկանալ, որ հոմանիշ բառերի դեպքում մի վեկտորը մյուսով փոփոխելիս՝ թարգմանությունը չի փոփոխվում, իսկ տառերին վեկտորներ համապատասխանեցնելիս չենք կարող գտնել հոմանիշ կամ իմաստով նման բառեր: Երկրորդ տարբերակն է ամեն բառին համապատասխանեցնել մի վեկտոր: Այս դեպքում երկու խնդիրն էլ լուծվում են: Ե՛վ մուտքային տվյալների քանակն է քիչ լինում, և՛ ցանցը կարող է սովորել նման բառեր: Բայց այս դեպքում ուրիշ խնդիր է առաջանում: Մեր ուսուցանման տվյալները սահմանափակ են, և թեստավորման ընթացքում հնարավոր է մուտքային բառ ունենանք, որը չկար ուսուցանման տվյալների մեջ: Այդ դեպքում ինչ վեկտոր պետք է համապատասխանեցնենք այդ բառին: Դա լուծելու համար օրինակ կարող ենք այն բառերին, որոնք առկա չեն եղել ուսուցանման տվյալների մեջ, վերագրել զրոյական վեկտոր: Բայց դա կարող է շփոթություն առաջացնել ցանցում: Դրա համար ծանոթանանք բառարան կազմելու երրորդ տարբերակին: Սահմանենք սկզբնական բառարանի չափ: Այն բառերը որոնք շատ են հանդիպել, կլինեն բառարանի անդամ: Հետո վերցնենք ամենաշատ հանդիպած մասնիկները: Այսինքն ”ուշադրություն” բառի մեջ ունենք ”ություն” մասնիկը, որը հայերենում շատ հաճախ է հանդիպում և կարող ենք այն նույնպես դարձնել բառարանի անդամ: Ամենակարևորը բառարանի մեջ պետք է ընդգրկենք բոլոր թվանշանները, բոլոր տառերը (մեծատառ, փոքրատառ), որպեսզի պատահական տեքստի դեպքում (”ասյկդհ345յաս”), այն նույնպես կարողանանք բաժանել մասերի (տառերի) և ամեն մասի համապատասխանեցնենք one-hot վեկտոր: Երբ մուտքում ունենանք ”ուշադրություն” բառը, սկզբից ստուգելու ենք, արդյո՞ք այն ամբողջությամբ առկա է բառարանում: Եթե ոչ, ապա դիտարկելու ենք նրա մասերը ([”ուշադրությու”,

”ն”], [”ուշադրությ”, ”ուն”], …, [”ուշադր”, ”ություն”])։ Վերցնելու ենք այն տարբերակը, որի դեպքում մասերից մեկը ունի իրեն համապատասխան վեկտոր և այն ամենամեծն է։ Այսինքն եթե և՛ ”ուշադրությու”, և՛ ”ուշադր” մասնիկները չունեն իրենց համապատասխան վեկտորները, բայց և՛ ”ուն”, և՛ ”ություն” մասնիկները ունեն համապատասխան վեկտորներ՝ վերցնելու ենք դրանցից ամենաերկարը(”ություն”)։ Եթե բառարանի մեջ ունենք ”ություն” մասնիկը, դրան համապատասխանեցնելու ենք one-hot վեկտոր և անցնելու ենք ”ուշադր” մասնիկին վեկտոր համապատասխանեցնելուն։ Նորից պետք է գտնենք ամենամեծ մասնիկը։ Այս դեպքում հնարավոր է ամենամեծ մասնիիկը մի տառ լինի և դրանից հետո ”ուշադր” մասնիկը բաժանելու ենք տառերի և ամեն տառին վերագրենք one-hot վեկտոր։ Արդյունքում ”ուշադրություն” բառը բաժանեցինք 6 մասնիկի (6 one-hot վեկտոր)։ Նախադասությունը մասնիկների բաժանելու գործընթացը կոչվում է tokenization, իսկ այն մոդելը, որի միջոցով բաժանում ենք մասերի՝ tokenizer: Ստացված մասերը կոչվում են token-ներ։ Տարբեր tokenizer-ների կարող եք ծանոթանալ tiktoken կայքում։ Այս կայքում gpt4 և gpt4-o մոդելները փորձարկելիս կարող ենք տեսնել էական տարբերություն հայերեն նախադասությունները tokenization կատարելիս։ gpt4 tokenizer-ը բաժանում է տառ առ տառ, իսկ gpt4-o tokenizer-ը բաժանում է ըստ մասնիկների։ Կարող եք ծանոթանալ gpt4 և gpt4-o tokenizer-ի բառարաններին՝ gpt4, gpt4-o։ gpt4 tokenizer-ի բառարանը բաղկացած է 100 000 token-ից, իսկ gpt4-o tokenizer-ի բառարանը 200 000 token-ից։
Այսպիսով տեքստերի հետ կապված խնդիրներում մեզ անհրաժեշտ է string-ը դարձնել թիվ (վեկտոր)։ Մեծ քանակի տվյալների վրա կառուցում ենք tokenizer, որը մուտքում ստանալով նախադասությունը, բաժանում է մասնիկների և ամեն մասնիկի բառարանից համապատասխանեցնում է one-hot վեկտոր։