ქართული ენის მოდელირების მცდელობა

კოტორეიშვილი ვახტანგი, კურცხალია დაჩი, სტურუა საბა
[vkoto17, dkurt17, sstur17]@freeuni.edu.ge
Free university of Tbilisi
Tbilisi, Georgia

აბსტრაქტი

ნაშრომში განხილულია ქართული ენის მოდელირების მცდელობა. დავიწყეთ მონაცემების შეგროვებით, მათი გაანალიზებითა და ვიზიალუზაციით. შემდგომ გადავედით დამუშავებულ მონაცემებზე სხვადასხვა ტიპის მოდელის აგებაზე. დავიწყეთ N-Gram-ით, რომელიც სიტყვათა მიმდევრობების სიხშირეებს იყენებს მოდელირებისთვის. მოცემულმა მიდგომამ ვერ შეძლო ქართული ენის სპეციფიკის გადმოცემა. ამის შემდეგ ვცადეთ უფრო კომპლექსური ტრანსფორმერ მოდელები, შევქმენით ვორდ ემბედინგები, გამოვიყენეთ სხვადასხვა ტიპის ტოკენიზაცია. საბოლოოდ კი ბევრი რამ გავიგეთ მოცემული ამოცანისა და მისი სირთულეების შესახებ.

შესავალი

ჩვენი ამოცანაა ქართული ენის მოდელირება. ენის მოდელის კონცეფცია დიდი ხანია არსებობს კომპიუტერულ მეცნიერებაში. დაწყებული სტატისტიკური მოდელებიდან, დამთავრებული ტრანსფორმერებით ბევრნაირადაა შესაძლებელი მოცემული ამოცანის გადაჭრა.

გამოცდილებამ გვაჩვენა, რომ კარგი ენის მოდელის შექმნას ჭირდება ენის დიდი კორპუსი და ასევე დიდი გამოთვლითი რესურსები. მაგალითისთვის, Open AI-ს ენის მოდელს gpt-3-ს აქვს 175 მილიარდი პარამეტრი. ამ კუთხით, შეიძლება ითქვას, ქართული ენა დაჩაგრულია. ჩვენ low resource ენებში გავდივართ და ამას ისიც ემატება, რომ ამ პროექტის განმავლობაში არ გვქონია ნორმალური GPU სერვერი (Kaggle-ს ჯიფიუს ვიყენებდით)

როდესაც ამოცანაზე ფიქრი დავიწყეთ, მივხვდით, რომ რამდენიმე გამოწვევის წინაშე ვიდექით: დავიწყოთ იმით, რომ გამოთვლითი რესურსების სიმცირის გამო, მოგვიწია მცირე ზომის კორპუსის გამოყენება; ამასთანავე, რადგან ქართულ ენაში არ გაგვაჩნია ხელსაწყოები, რომლითაც შევძლებთ სიტყვის შედარებით მარტივ ფორმაზე დაყვანას (მაგალითად ფუძის გამოყოფა რომ შეიძლებოდეს) შედეგად მივიღეთ ძალიან ბევრი უნიკალური ტოკენი, რომელთა სიხშირეც საკმაოდ მცირე იყო; ამას ემატებოდა ისიც, რომ თითქოს ყველაზე მიმზიდველი კორპუსი - ვიკიპედია - შეიცავს სპეციფიკურ ტექსტებს და, ჩვენი აზრით, კარგად არ ასახავს ქართულ ენას (ტოპ 10 ყველაზე ხშირ სიტყვაში შედის “წელი” და “წლის”);

მიუხედავად ზემოხსენებული გამოწვევებისა, ვეცადეთ ტრივიალური მიდგომით დაგვეწყო და ნელ-ნელა “გაგვერთულებინა”, პროცესში გვესწავლა რას ვაკეთებდით არასწორად და როგორ შეიძლებოდა მიღებული შედეგის გაუმჯობესება. ამ ყველაფერზე უფრო ვრცლად მომდევნო თავებში დავწერთ.

მონაცემების შეგროვება

როგორც ვახსენეთ, ენის მოდელს დიდი რაოდენობით ტექსტური მონაცემი ესაჭიროება. ვცადეთ დაკავშირება ქართული ენის ეროვნული კორპუსის საიტთან, თუმცა პასუხი არ გაგვცეს. კორპუსის პოვნის მცდელობა ნაკლებად რელევანტური გახდა მის მერე, რაც გავიაზრეთ, რომ ნორმალურ GPU სერვერს ვერ ვიპოვიდით. საბოლოოდ, სხვა გუნდებთან ერთად შევჯერდით ვიკიპედიას კორპუსზე, რომლის მოპოვებაც საკმაოდ მარტივი გამოდგა. თუმცა, მოდელის გასატესტად გავპარსეთ marketer.ge.

მონაცემთა ანალიზი და პრეპროცესინგი

როგორც შესავალში აღვნიშნეთ, კორპუსი არ შეესაბამება ქართული ენის ზოგად სახეს. ვიკიპედიას სტატების სპეფიციკიდან გამომდინარე, გვხვდება მეტი სასვენი ნიშანი. ამავდროულად, 10 ყველაზე ხშირ სიტყვაში შედის ისეთი სიტყვები, როგორიცაა “წელი” და “წლის”, რაც ნამდვილად არ არის კორექტული ქართული ენისთვის.

ტექსტის პრეპროცესინგის პირველი და ტრივიალური მიდგომა იყო უბრალოდ სასვენი ნიშნების გამოყოფა და “სფეისების” მიხედვით გაყოფა. ამ მიდგომით დავთვალეთ უნიკალური ტოკენები, ავარჩიეთ ტოპ 80-90 ათასი, რომელიც შევიყვანეთ ჩვენს vocabulary-ში და დანარჩენს მივანიჭეთ unk ტოკენი.

მოცემული პრეპროცესინგის შედეგად მივიღეთ ძალიან ბევრი unk ტოკენი, რაც, თავისთავად, პრობლემაა. სამწუხაროდ, ეს პრობლემა ვერ მოგვარდებოდა vocabulary-ს ზომის გაზრდით, რადგან კორპუსი მცირე ზომის გვქონდა, vocabulary-ს გაზრდა კი ავტომატურად ნიშნავდა მოდელის პარამეტრების ზრდას და ისეთი სიტყვების გათვალისწინებას, რომლებიც იშვიათად გვხვდება ისედაც მცირე ზომის კორპუსში. ჩვენი აზრით, vocabulary-ს გაზრდა ვერ მოაგვარებდა ზემოხსენებულ პრობლემას. შესაბამისად, გადავწყვიტეთ ყურადღება გადაგვეტანა პრეპროცესინგის ნაწილზე.

პრეპროცესინგის მეორე მცდელობისას ვეცადეთ unk ტოკენში უფრო მეტი ინფორმაცია ჩაგვედო. პირველი მცდელობა ერთსა და იმავე შემთხვევად განიხილავდა უცხოურ სიტყვას, ქართულ უცნობ სიტყვას, რიცხვს და ა.შ. შევქმენით სხვადასხვა ტიპის unk ტოკენები : ციფრები (NUM_SML), რიცხვები 10დან 2100მდე (NUM_MED), რიცხვები 2100დან (NUM_LRG), უცნობი სიმბოლო (SYMB_UNK), უცნობი უცხოური სიტყვა (FOREIGN_UNK), უცნობი ქართული სიტყვა (FOREIGN_UNK).

გარდა ნახსენები პრეპროცესინგისა, ვცადეთ subword ტოკენაიზერის (BPE Tokenizer) გამოყენება, რამაც ბევრად უკეთესი შედეგი მოგვცა. ეს ბუნებრივია, რადგან აქამდე თუ ორი მსგავსი სიტყვა, მაგალითად “ადამიანი” და “ადამიანები” ორ სხვადასხვა ტოკენს წარმოადგენდა, subword მოდელირებით შესაძლებელია ამ ორ სიტყვას საერთო ტოკენი/ტოკენები ჰქონდეს.

მოდელის აგება

სანამ ძირითადი მოდელის აგებას დავიწყებდით, ავაგეთ მარტივი N-gram მოდელი. N-gram[1] მოდელი გულისხმობს ტექსტის სტატისტიკურ ანალიზსს შემდეგნაირად: ვითვლით ერთად შემხვედრი სიტყვათა ჯგუფების რაოდენობას. შემდეგ ალბათობების შეწონვისთვის გამოვიყენეთ Laplace Smoothing. იქიდან გამომდინარე რომ N-gram მოდელი გულისხმობს ყველა ნ-ჯგუფი ტოკენების შენახვას, პრობლემა შეგვექმა მეხსიერების მხრივ. ამატომ მოდელი დავატრენინგეთ მონაცემთა მესამედზე. როგორც მოსალოდნელი იყო, "მოდელის" მიერ შექმნილი წინადადებები სატრენინგო დატაზე იყო მთლიანად მორგებული და თავისით ვერ იგონებდა ახალ, გამართულ წინადადებებს. ამას დაემატა ისიც, რომ მცირე ზომის კორპუსის გამო, მოგვიწია N მცირე აგველო და, შესაბამისად, დაგენერირებულ ტექსტს არ გააჩნდა კონტექსტი.

გადავწყვიტეთ, უფრო კომპლექსური მიდგომა გვეცადა. ჩვენთვის ძალიან საინტერესო იყო ტრანსფორმერ არქიტექტურა, რაც დღეს SOTA-დ ითვლება. შესაბამისად, დავიწყეთ ყველაზე მარტივი ვერსიით, ანუ არქიტექტურით, რომელიც ნაშრომში - Attention is all you need - იყო შემოთავაზებული.[2]

ტექტის გენერაციისთვის გამოვიყენეთ Beam Search ალგორითმი. ბიმ-სერჩის დროს ტექსტის გენერაციის დროს ვაკვირდეით N ყველაზე მაღალქულიან შედეგს, რომელსაც შემდეგი სიტყვის გენერაციის დროს ვიყენებთ. ყოველ ნაბიჯზე "ვსხლავთ" და ვტოვებთ მხოლოდ N მაღალქულიან შედეგებს. ბიმ სერჩის გარდა გამოვიყენეთ greedy და რენდომ გზებიც თუმცა ვფიქრობთ ბიმ სერჩი ყველაზე გამოსადეგია.

გადავწყვიტეთ მოდელს "დავხმარებოდით" და შევქმენით ვორდ ემბედინგები word2vec ალგორითმის მიხედვით. ქვემოთ მოყვანილი მოდელები დავატრენინგეთ როგორც pretrained ემბედინგებით, ასევე მათ გარეშე, თუმცა რეალური სხვაობა შედეგში არ ქონია. ემბედინგებზე მეტი ინფორმაციისთვის წაიკითხეთ ნოუთბუქი - Word2vec Visualization and Analysis.ipynb.

პირველი მოდელი დავატრენინგეთ პირველი პრეპროცესინგით დამუშავებულ ტექსტზე შემდეგი პარამეტრებით: ემბედინგის განზომილება - 100; ენკოდერში ჰიდენ ლეიერის ზომა - 200; ენკოდერ ლეიერების რაოდენობა - 2; attention-ების რაოდენობა - 2; დროფაუთი - 0.3; Cross Entropy ობჯექთივ ფუნქცია და SGD ოპტიმაიზერი. შედეგად მივიღეთ მოდელი, რომელიც ძალიან დიდ უპირატესობას ანიჭებდა unk ტოკენს. ეს ბუნებრივიც იყო გამოყენებული პრეპროცესინგიდან გამომდინარე.

როდესაც იგივე მოდელი მეორე ტიპის პრეპროცესინგით დამუშავებულ ტექსტზე ვატრენინგეთ, უკეთესი, მაგრამ მაინც ცუდი შედეგი მივიღეთ. მოდელი ისევ უპირატესობას ანიჭებდა unk ტოკენებსა და ამჯერად უკვე სასვენ ნიშნებსაც. ამის მერე, ვივარაუდეთ, რომ პრობლემა კორპუსში იყო: გვხვდებოდა დიდი რაოდენობით სასვენი ნიშნები და სიტყვების უმეტესობა მცირე რაოდენობით იყო წარმოდგენილი.

---

[1] Daniel Jurafsky & James H. Marti. Speech and Language Processing.- Chapter 3- N-gram models
[2] Ashish Vaswani Noam Shazeer Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin - Attention is All You Need

ზემოხსენებული პრობლემის გადაჭრა ვცადეთ ოპტიმაიზერის შეცვლით. SGD-ს მაგივრად, არჩევანი შევაჩერეთ Adagrad-ზე[3]. SGD-სგან განსხვავებით, Adagrad-ი განსხვავებულ ლერნინგ რეითს ურჩევს თითოეულ პარამეტრს თითოეულ თრეინინგ ნაბიჯზე. ეს კი იმაში გვეხმარება, რომ ხშირი სიტყვები დაბალი ლერნინგ რეითით შეიცვლებიან, ხოლო იშვიათები - დიდით. სამწუხაროდ, არც ეს ცვლილება გამოდგა მნიშვნელოვანი და მიღებულ მოდელს ისევ უჭირდა ადეკვატური ტექსტის დაგენერირება.

გამომდინარე იქედან, რომ გვქონდა მცირე გამოთვლითი და დროითი რესურსები, გადავწყვიტეთ შეგვეცვალა მიდგომა. Huggingface ბიბლიოთეკის გამოყენებით შევქმენით subword ტოკენაიზერი და Roberta-ს არქიტექტურის მოდელი ვატრენინგეთ ვიკიპედიას კორპუსზე[4]. ამ მიდგომამ ბევრად უკეთესი შედეგი მოგვცა, თუმცა იგი მაინც შორსაა სასურველისგან. მოდელი მაინც უპირატესობას ანიჭებს სასვენ ნიშნებს და დაგენერირებული ტექსტი კარგად არ ასახავს ქართულ ენას.

ევალუაცია

ევალუაციისთვის გამოვიყენეთ ქართული ვებსაიტის - Marketer.ge-დან წამოღებული ქართული სტატიები. მარკეტერის სტატიები ვიკიპედიის სტატიებისგან განსხვავებული ლექსიკით არის დაწერილი. შესაბამისად ევალუაციისას მიღებული განსხვავებები და შედეგი კარგი დასაკვირვებელია. იქიდან გამომდინარე, რომ მოდელის ტრენინგმა პრობლემები შეგვიქმნა, საშუალება არ მოგვეცა რომ ევალუაციისას კარგად შეგვესწავლა მოდელის ქცევა.

დასკვნა

მოცემული კვლევით ფეხი შევდგით იმ გაუკვალავ ტყეში, რომელსაც ქვია ქართული ენის მოდელირება. ამოცანა, მართლაც რომ, რთულია, თუმცა ვფიქრობთ, რომ დღევანდელი NLP-ს ტექნიკებით სავსებით შესაძლებელია კარგი ენის მოდელის შექმნა.

დავიწყეთ ტრივიალური მიდგომით, ვცადეთ დაგვენახა პრობლემები, შეგვეცვალა მიდგომა და ასე გაგვეუმჯობესებინა შედეგი, მაგრამ სასურველი შედეგის მიღება ჯერჯერობით ვერ შევძელით.

ნათლად დავინახეთ ის, რომ კარგი ენის მოდელის შექმნას ესაჭიროება დიდი რაოდენობით მონაცემები და გამოთვლითი რესურსი. ჩვენი აზრით, სწორ გზაზე ვართ და თუ მეტ ყურადღებას დავუთმობთ ენის კარგი კორპუსის შექმნას, საკმაოდ კარგი შედეგი გვექნება.

ბიბლიოგრაფია

[3] An overview of gradient descent optimization algorithms

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov - RoBERTa: A Robustly Optimized BERT Pretraining Approach

- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov - RoBERTa: A Robustly Optimized BERT Pretraining Approach
- Ashish Vaswani Noam Shazeer Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin - Attention is All You Need
- Daniel Jurafsky & James H. Marti. Speech and Language Processing.- Chapter 3- N-gram models
- An overview of gradient descent optimization algorithms