# Georgian Language Modelling

**Shota Elkanishvili  Otari Emnnadze**

## Abstract

We have trained popular NLP models on Georgian Language and compared the results on text generation task.

To address the issue, we tried a range of model types, from N gram to BERT. All models proved to be suitable in terms of the computational constraints.

As a result, you can download pretrained models and made them work out on downstream tasks, like text classification, entity recognition and etc. We also published the training codes, so you can also train models from scratch.

## 1 Introduction

Large corporations do not have separate (monolingual) language models for Georgian since it is seen as a low resource language and because so few people speak it. The fact that there aren't many Georgian ML engineers interested in creating language models for the Georgian language is another reason why NLP problems for the Georgian language aren't fully investigated.

That is why we decided to create language models for general purpose embeddings and then made some text generation models from them.

While building this model, we had to solve several important tasks:

- In contrast to the English language, the corpus of the Georgian language is not sufficiently large. There is no public clean language corpus, for experimenting, so we had to manually search for unclean data. Another problem related to data, is that, corpus quality is not controlled, so we had to made all of the preprocessing in order to get more or less good quality data for training.
- Since the Georgian language is less well-known in the realm of NLP than other languages, we lacked suitable word vectors. Due to this, we were required to complete an independent work that involved training Word2Vec on the bigger dataset and evaluating it by checking similarity and analogies over the words.
- Unlike English and other high resource languages, there are no benchmarks for Georgian language. So we had to manually create n-gram model to check the quality of other generator models, by comparing them to this baseline.

We trained LSTM model using our trained Word2Vec embeddings, which was frozen in the training. Lstm did not do well in the text generation because Word2vec embeddings were trained on word level

and not on sub-word level, resulting in a vast search space. We also trained BERT with the MLM (Masked Language Modelling) task, and we used the last layer of the model with the beam search algorithm to generate the next word for the given context.

## 2 Data

We searched for large Georgian language corpus, but only things we found were either too small or too noisy. Finally we chose OSCAR dataset because it contained almost every article from almost all the Georgian websites.

This corpus is used in many multilingual model training papers.

## 3 Data analysis

Because the data was mostly collected on the internet, it had noise which would only confuse the model. Because of that we started cleaning the data.

We started with cleaning all the words which included any characters except Georgian letters and numbers, then we removed punctuation.

After the analysis of individual word (Fig 1-2) lengths in sentences, we found out that there were some irregularities, so we cleaned every word which contained 26 or more letters(Fig 3), because most of the time these were sentences written without a space in between.

Third step was to study sentences, we plotted sentence lengths and filtered every sentence which contained under 5 and above 500 words. Also we made character level analysis (Fig 4)

Figure 1

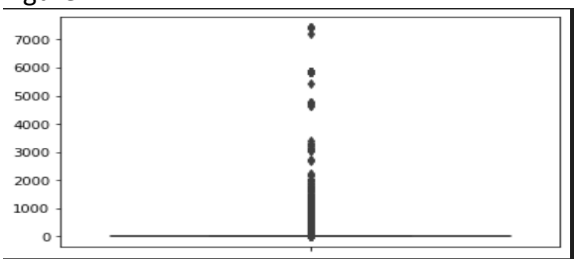



Figure 2

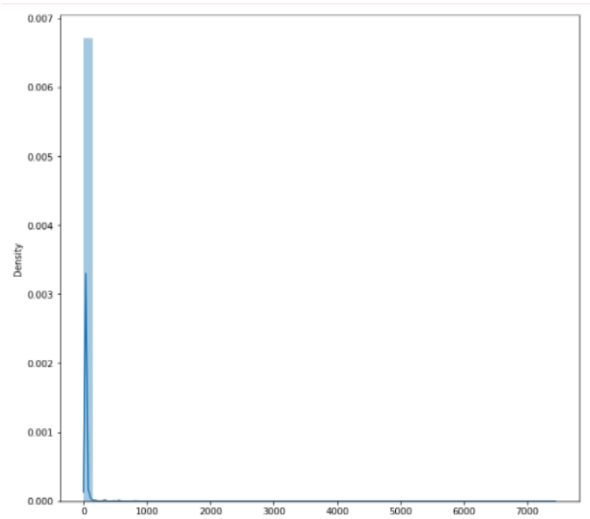



Figure 3

```
Counter({1: 3545082,
         2: 15585495,
         3: 7748880,
         4: 13849053,
         5: 14500064,
         6: 17305368,
         7: 18253932,
         8: 17711760,
         9: 15370712,
         10: 12616823,
         11: 9809304,
         12: 6701230,
         13: 3540695,
         14: 2065788,
         15: 1355441,
         16: 653189,
         17: 334616,
         18: 167514,
         19: 122840,
         20: 74155,
         21: 82537,
         22: 41283,
         23: 18116,
         24: 14336,
         25: 16140})
```

Figure 4

```
[('6', 16763),
 ('8', 16781),
 ('7', 19772),
 ('ჟ', 20300),
 ('4', 20993),
 ('3', 21376),
 ('9', 23029),
 ('2', 26129),
 ('5', 27546),
 ('1', 31894),
 ('0', 34568),
 ('ჭ', 36401),
 ('ჳ', 47095),
 ('ჲ', 54234),
 ('ძ', 72320),
 ('ღ', 80611),
 ('ჩ', 97490),
 ('ყ', 102380),
 ('წ', 132823),
 ('ქ', 158055),
 ('ფ', 163832),
 ('ვ', 177703),
 ('ზ', 196429),
 ('ხ', 226896),
 ('ც', 265065),
 ('ძ', 273892),
 ('ჯ', 363821),
 ('ტ', 442098),
 ('ბ', 487274),
 ('თ', 511396),
 ('ს', 586003),
 ('დ', 693272),
 ('გ', 694653),
 ('ლ', 821854),
 ('ბ', 846055),
 ('ი', 875075),
 ('ნ', 984662),
 ('ო', 1041996),
 ('რ', 1056997),
 ('ს', 1222632),
 ('ე', 1803297),
 ('ი', 2228885),
 ('ა', 2775397)]
```

After cleaning the corpus, data became balanced and it satisfied all the criteria for language models to learn the Georgian language.

# 4 Models

Because we did not have enough computational power and already created benchmarks, we had to do everything manually and also we had limitations in choosing parameter sizes inside models. We used following models:

## 4.1 N Gram Model

First model that we trained was N gram, for implementation we used NLTK library and MLE (maximum likelihood estimator) class. We tried different n for gram size, and finally best one turned out to be 3.

Because of the computational and memory limitations, we only used half of the data for training, which was 5 million sentence and almost 70 million tokens.

Results were really impressive, for example on input “ქართველი ერი ამას ვერ აიტანს” model continued text with “რომ სირცხვილი აჭამოს ვინმემ”. Which was nice completion.

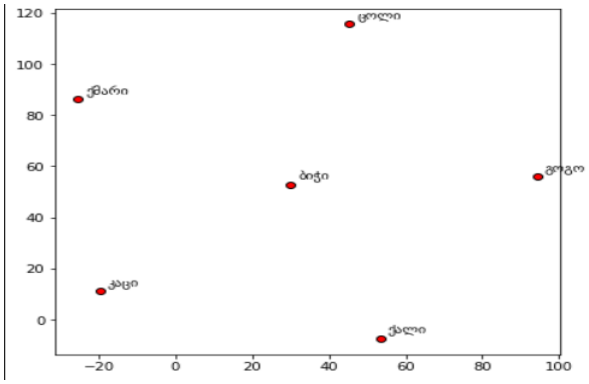
## 4.2 Word2Vec

Second model we trained was Word2Vec, which maps word to the latent space embeddings. These embeddings are learnt that way that, they contain relationship-like information between words, which helps engineers to develop other tasks using this pretrained embeddings.

For implementation, we used Gensim library, and we trained model on the whole data, which included 9 million sentences and 167 million tokens. For embedding size, we chose 100, because we had memory and computational limitations.

We evaluated model, by checking the same characteristics as original paper model. We used cosine similarity to search for the closest words by meaning, and then we searched for analogies, which was originated in the original paper.

Figure 5



We used T-SNE dimensionality reduction method to reduce 100 numbers to two, and then plot words on the axes. We chose 6 words: ქმარი(husband), ცოლი(wife), ბიჭი(boy), გოგო(girl), კაცი(man), ქალი(woman), used Word2Vec embeddings and reduced them to two, and plotted them. If we take a close look to a plot we will find out that, husband-man vector is parallel to wife-woman vector. Husband – boy vector is parallel to wife-girl pair, and boy-man pair is parallel to girl-woman pair. With these examples we can safely say that our model trained well, and it was capable of extracting information from words by using skip-gram method.

Figure 6

```
most_similar(model,'საქართველო')
```

```
[('რუსეთი', 0.7153475284576416),
 ('ქვეყანა', 0.7017509341239929),
 ('ევროპა', 0.6886364817619324),
 ('საქართველოც', 0.6679713129997253),
 ('სომხეთი', 0.6543735861778259)]
```

Figure 7

```
analogy(model,'დილა','8','24')
```

'საღამო'

```
analogy(model,'სამი','ორი','ოთხი')
```

'ხუთი'

```
analogy(model,'სამი','ორი','ხუთი')
```

'ექვსი'

```
analogy(model,'საფრანგეთი','პარიზი','ამსტერდამი')
```

'ჰოლანდია'

By looking figure 6 and 7 we can clearly see how good our model is, in the similarity example, query is Georgia, and most similar words are Russia, Country, Europe and Armenia, all of these words are relevant, for that query. On analogy side we have morning-8+24 = evening, three-two + four = five, three – two + five = six, France-Paris + Amsterdam = Netherlands. With these analogies we can see, that model has meaningful vectors, which contain many relationship-like information.

## 4.3 LSTM

At this stage, we wanted to use word embeddings and generate text with context. LSTM is variation of recurrent neural network, which deals with auto regressive sequence kind of inputs, where the alignment matters.

For implementation, we used Keras library. We defined input as a five word sequence, and output was the sixth word in the sentence. We developed keras sequential model, where first layer was frozen embedding layer, filled with our trained Word2Vec vectors. Second layer was LSTM layer, then the dense layer to generate samples and finally the softmax layer to make multi class classification.

Model turned out to be large, because vocab size was half a million tokens, so model had to predict which token was the best fit from that set. Model's complexity was making everything hard, it studied that the best way to make lose as small as possible was to always predict the most frequent word – 'და'- which means "and".

Best way to solve this issue would be to train Word2Vec on sub-word level tokens and then train lstm model the same way.

It would dramatically reduce the output layer size and it would make training process easier for model, and it would not stuck at a very bad local optimum.

## 4.4 BERT

Transformers dominated the NLP world, paper "attention is all you need", is said to be the "image net moment", of the NLP community. Second model which changed the way we looked at NLP problems was BERT. Because of that huge capabilities BERT model had, we wanted to train it on Georgian language corpus.

For Implementation we used Transformers library from Hugging Face, we trained our own wordpiece tokenizer, and trained BERT with 200 000 sentences on MLM (Masked Language Modelling) task, we trained it on 20 epochs.

For inference, we used BERT with the last layer, which is responsible for MLM prediction. We added mask token as a last word and asked BERT to predict it. For completing the sentence, we compared greedy and beam search algorithms. Beam search worked much better, but because mainly in the community MLM is not used for text generation, beam search also did not help that much to make the perfect model.

# 5 Conclusion

Finally, we can say that we created very good Georgian language models for getting context based or word based embeddings. We also created very good not deep learning-based text generation model and BERT model which is also good at text generation.

If we had more computational power, we would train much more sophisticated models, with better capacity of generating Georgian sentences.

## 6 References

Word2Vec: Word2Vec original paper

Transformers: Transformers Original Paper

BERT: BERT original paper

Material that helped on implementation side:

bert pretraining

how to train bert