Final NLP class Project Report

BBC News Analysis

Mahsa Ghaderan

Supervise by

Dr Sauleh Etemadi

July 17, 2021

Department of Computer Engineering

Iran University of Science and Technology

Contents

List of Figures

List of Tables

Part 1

Word2Vec

In this part I used assignment2 from cs224n, 2021 Stanford NLP course as my base code. Different Word2Vec models are trained for each label which exist in BBC data set. Headline is concatenated with body of each news for as input text for the model. Each models trains for 40000 iteration. It took about 5 hours for each label. Wights of models are saved is "models/word2vec" directory. Most 30 repeated words are chosen from each label. Images - , - , - and - shows distribution of each model on a 2D map.

1.1 Iran News

After training word2vec model we exoect to see words with same meaning appear near to ech other and words with far meaning and context place far from others on each map.

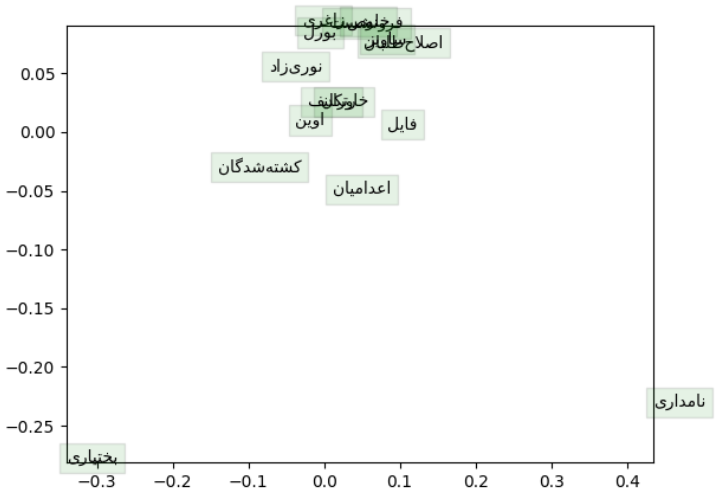

Figure 1.1: Sub-words vocab example

1.2 Art News

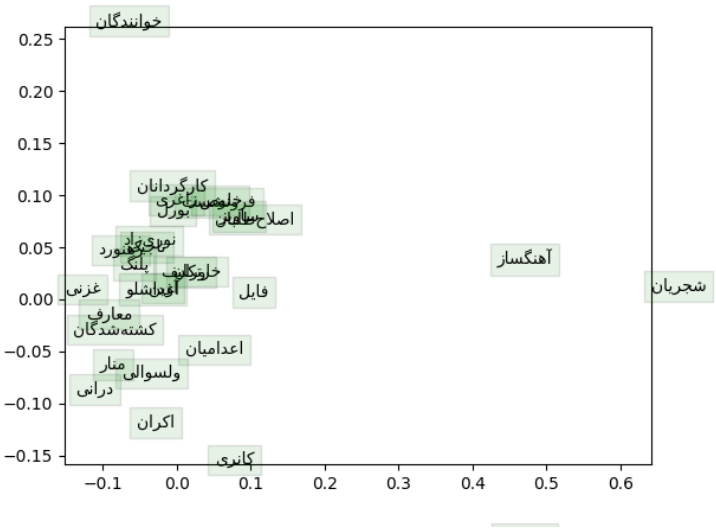

Figure 1.2: Sub-words vocab example

1.3 Sport News

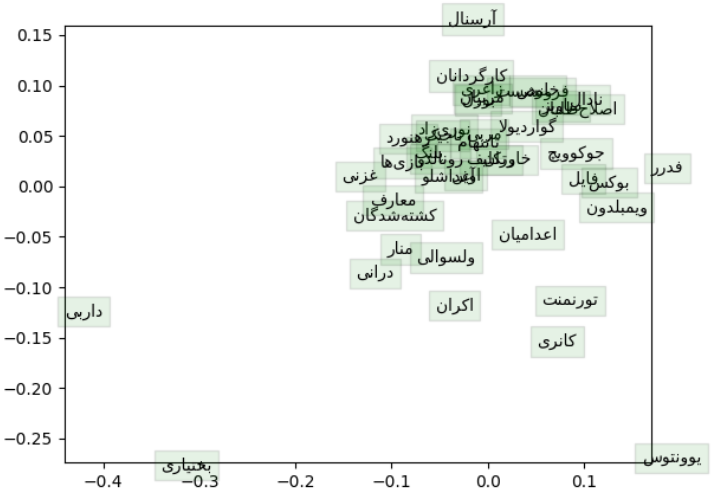

Figure 1.3: Sub-words vocab example

1.4 Economic News

In map 1.4 words in car and it's factory are gathered in the left part of the image.

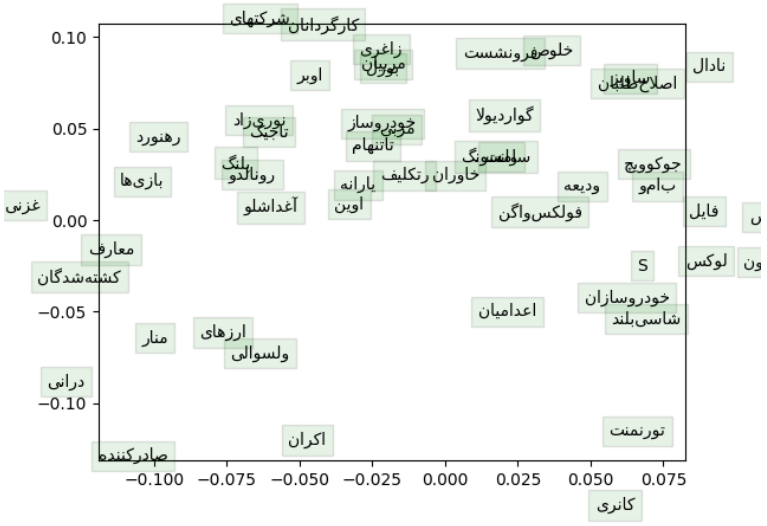

Figure 1.4: Sub-words vocab example

1.5 Science News

At map 1.5 words are overlapped and it's not easy to distinguish them. Equivalent of two words Astronauts and UFO persia are in same context and located almost near to each other and far from others. Two other words are composer and name of a popular Iranian singer, Shajarian, are placed next to each other.

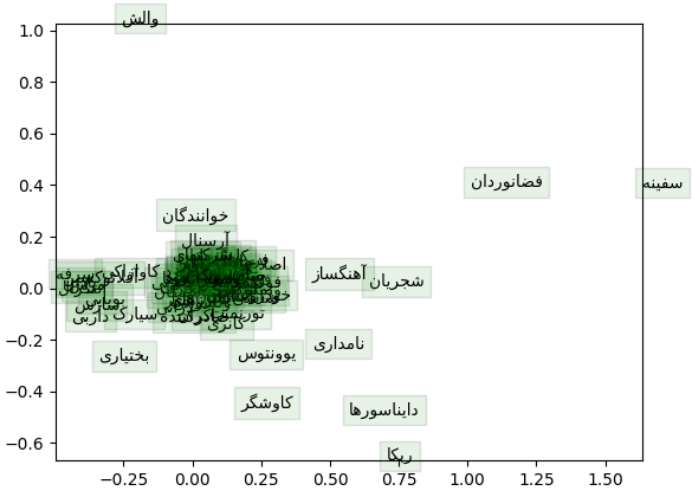

Figure 1.5: Sub-words vocab example

Part 2

Tokeniation

SentencePiece is an unsupervised text tokenizer and detokenizer python project, which mainly used for text generation tass.This part ia mainly based on this packet. It use a model to train what tokens are best to tokenizing a given corpus. In this part 2 different methods of tokenization in implemented. First is tokenizing words and sub-words, in this methods model looking for most repeated words and sub-words and even letter. As a result we can see almost every word in final version of chosen tokens. On the contrary the second method is just tokenizing words, this models looking for most repeated words.
Until this part entire headline and body of each news are both covered in the corpus but from this part English values are removed from corpus so tokenization and language generation will focus on Persian.
The corpus is shuffled 5 times and separated to train and test data. Also tokenization is applied on 4 different vocab size due to possible SentencePiece size and performance of output model is evaluated against these vocab size. In the following section, we discuss about effects of vocab size from very small value increasing to a large value.

2.1 Sub-word Level

At first i train tokenizer model on sub-words. It means model not only pays attention to words, but also pays attention to sub-words during training time.

When the vocab size is 1000, it is too small for such a corpus and due to model type, model focuce on smaller sub-words rather than meaning full words. In final choosed-vocab, almost every character in source language exists. So when we tokenize a text

there woudln't be any <unk> token and most words break into sub-words.
As vocab size increases model learns more complicated sub-words and less meaning-less characters and sub-words. Bot the negative point is as the vocab size increases it would be harder for models to learn task and the positive point is more context will be save after tokenization.

Another point is SentencePiece can discriminate between statrting token and not starting token and ending token of a word in this mode. As we can see in 2.1 the are different form of one token.

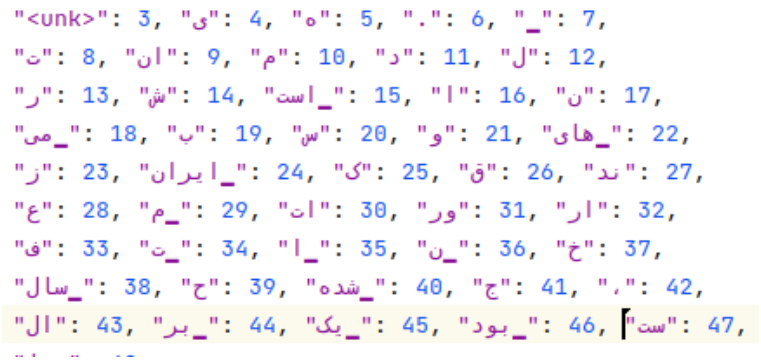

Figure 2.1: Sub-words vocab example

2.2 Word Level

In this part, i used word level tokenization, in this experience there exist <unk> token due to vocab size and model dealing with choosing best token in order to minimize <unk> token in unseen text.
Same as previous experiment, data-set is divided into train and test set and 5 different shuffled corpus is saved, this help us to have a better evaluation on each vocab size. Four different vocab size is defined as follow: 1000, 5000, 10000, 15000. Number of <unk> token and percent of that over each corpus and vocab is reported in "report/tokenization" directory and each model and output on test set is saved in "src/tokenization/working-dir/words-out". percent of each vocab size is presented in table 2.2.

	word-level					
corpus	1	2	3	4	5	AVG
1000	0.44	0.44	0.44	0.43	0.44	0.44
5000	0.20	0.20	0.20	0.20	0.20	0.21
10000	0.14	0.14	0.14	0.14	0.15	0.15
15000	0.12	0.12	0.12	0.12	0.14	0.13

As it is illustrated in table 2.2, number of <unk> token increase by vocab size reduction. the fewer <unk> token, the fewer information loss. On the other the larger vocab size leads into more amount of computaion for future task. As we saw it is a trade of between these to parameters, in my opinion best vocab size due to this corpus is **10000**. It has reasonable <unk> token and it may need less computation rather than 15000 vocab size.

Part 3

Language Model

Language model neural network artichecture is adopt from pytorch/examples project. Tokenizer codes and models are replaced with Part 2 and vocab for Language Modeling is construct in part 2. In this part the corpus is divided into three part, train set, dev set and test set. LanguageModel is trained on each label separately to evaluate capability of each model to generate specific news. Models are saved in "models/lm" directory. Models are trained on GTX1070 GPU. training process is 10 time faster than CPU.

3.1 LSTM

Use LSTM architecture(Figure 3.1) for language model. Best perplexity recorded is 47 on "Science" label. Each epoch last about 0.36 second, perplexity method starts from 60 and it is 43 on validation data on last epoch.3.2

In this trial, there are many <unk> token in output generated text. But generated sentences are in "Science" context.3.3

3.2 GRU

Use GRU architecture (Figure 3.4) for language model. Few Last epochs and output is illustrated in figure 3.5. LSTM and GRU have similar learning power as a evidence perplexity of GRU model on test set is 43, when label is "Science", same as LSTM. Each epoch last about 0.33 second as expected GRU model is faster, compare to LSTM model.

In this trial, again there are many <unk> token in output generated text. Gener-

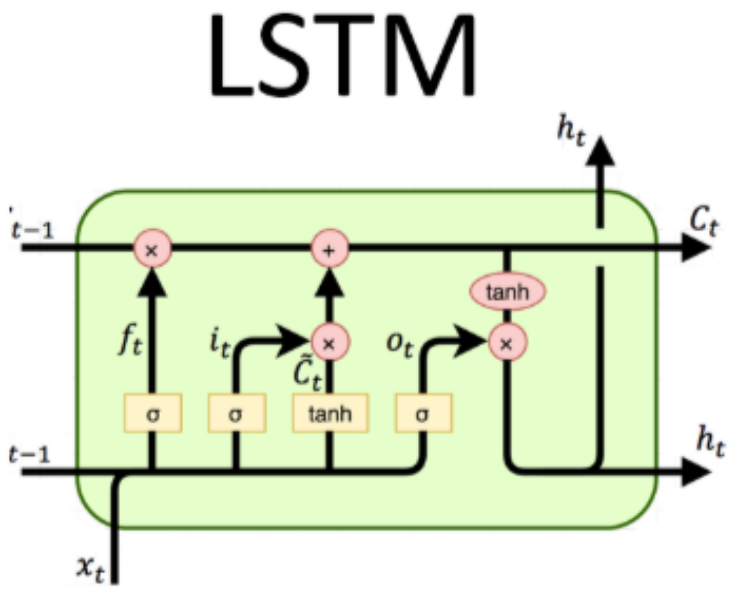

Figure 3.1: LSTM Architecture

```
| end of epoch  35 | time:  0.38s | valid loss  3.86 | valid ppl    47.48
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
| end of epoch  36 | time:  0.36s | valid loss  3.86 | valid ppl    47.47
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
| end of epoch  37 | time:  0.36s | valid loss  3.86 | valid ppl    47.46
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
| end of epoch  38 | time:  0.36s | valid loss  3.86 | valid ppl    47.45
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
| end of epoch  39 | time:  0.36s | valid loss  3.86 | valid ppl    47.42
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
| end of epoch  40 | time:  0.36s | valid loss  3.86 | valid ppl    47.42
-----------------------------------------------------------------------------
=============================================================================
| End of training | test loss  3.77 | test ppl    43.17
=============================================================================
```

Figure 3.2: Perplexity LSTM language model, label = Science

```
_است _شکست _دیگری _هنگام <unk> _برگزاری _اجرایی _بعد <unk> _افراد _با <unk> _سال <unk> <unk> <unk> _زمانی _کردن _داشته _همین
_جمعه <unk> <unk> <unk> _محیط _وزیر <unk> <unk> <unk> <unk> _خطر _دلار _چند <unk> _کمک _جان <unk> _بین _وبروس
_وجود _ماه _افراد <unk> <unk> <unk> _بازی <unk> _زندگی ۲۰۲۰_ unk> <unk> _ده _داشته _نظیر _واکسن _آیا _چند <unk> _دو
_فوق <unk> _شش _ایران _خیلی <unk> <unk> _همین _صورت <unk> <unk> _سال <unk> _سلامت _بود _دانشگاه _اولین _هستند.
_۶ خواهد _ساخته _کمی _فروردین _دو _آنها _درصد <unk> می _است. _بیش _گونه _کشور _ممنوع <unk> <unk> <unk> <unk> <unk>
دیگری _حال _دلار <unk> ویروس _زندگی _آزمایش _بالا <unk> <unk> نقش <unk> تاریخی <unk> <unk> <unk> اول <unk> <unk>
_یک <unk> _نیست _موارد _کرده _کند _تصویر <unk> <unk> _تغییر <unk> <unk> <unk> <unk> _یکی _شود. _کره <unk> _پیش _آزمایش
<unk> <unk> بهبود <unk> <unk> _نیمه <unk> پزشکی _نفر _همراه _درصد _تصویر <unk> بودند _بیرز <unk> جمع _چند _چند <unk> <unk>
<unk> <unk> _دانشگاه _است. _تعیین <unk> <unk> <unk> <unk> <unk> <unk> _بود. _مرکز _است. فاصله . <unk> <unk> <unk> _۵ نفر
```

Figure 3.3: Generated Text LSTM language model, label = Science

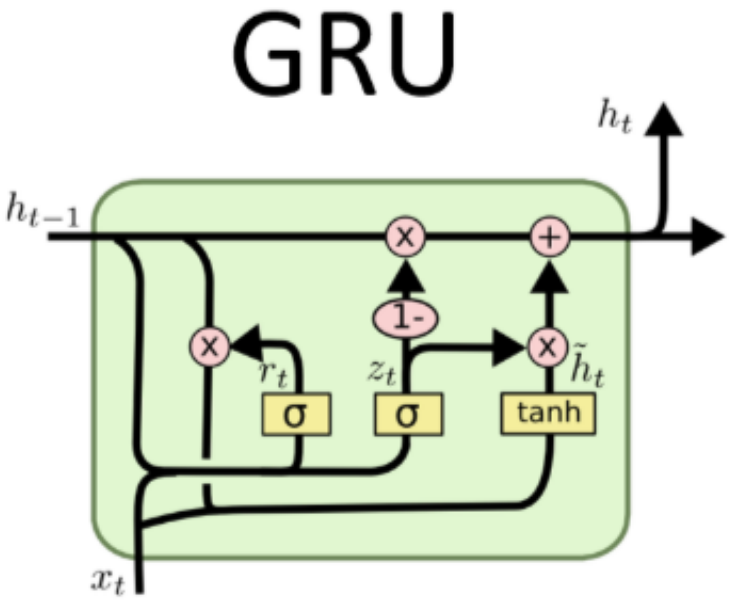

Figure 3.4: GRU Architecture

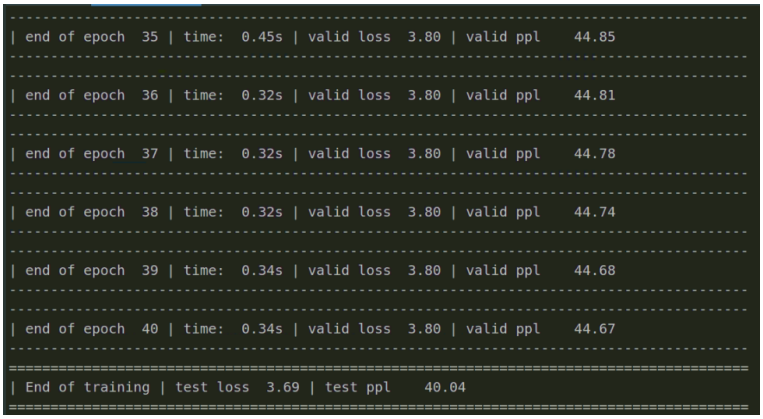

Figure 3.5: Perplexity GRU language model, label = Science

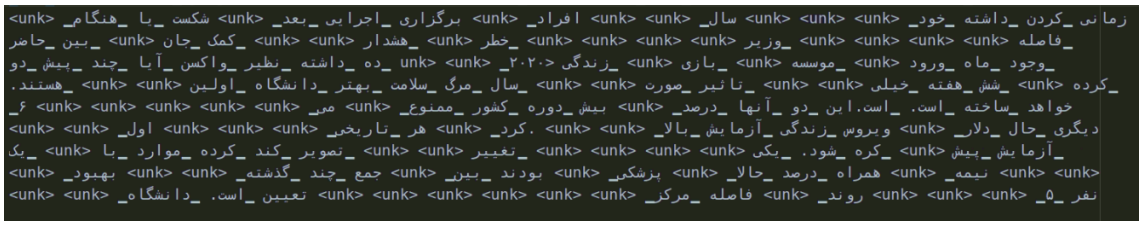

Figure 3.6: Generated Text GRU language model, label = Science

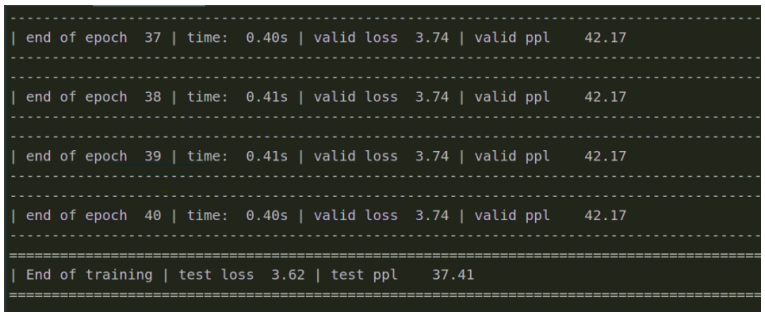

Figure 3.7: Perplexity Transformer language model, label = Science

ated sentences are in ”Education” context.3.6

3.3 Transformer

In last trial Transformer architecture is used for language model. Few Last epochs and output is illustrated in figure 3.7. LSTM and GRU both have almost similar learning power, but with transformer architecture it is possible to access better results. As shown in Figure 3.8 test perplexity is 37.41, when label is ”Science” which is 3 score better than previous models. Each epoch last longer than previous models and it is about 0.40 second.

In this trial generated text is shown in 3.8. Number of <unk> token is decreased and there are more meaningful sentences. Generated sentences are in ”Science” context.

Run ”bash run.sh 4” to train all three types of language models. Models will save

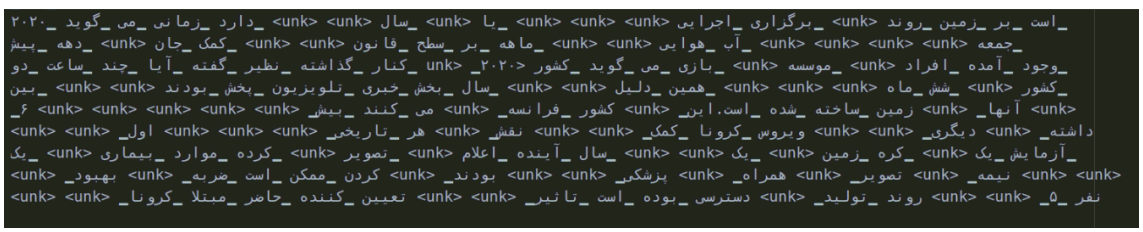

Figure 3.8: Generated Text Transformer language model, label = Science

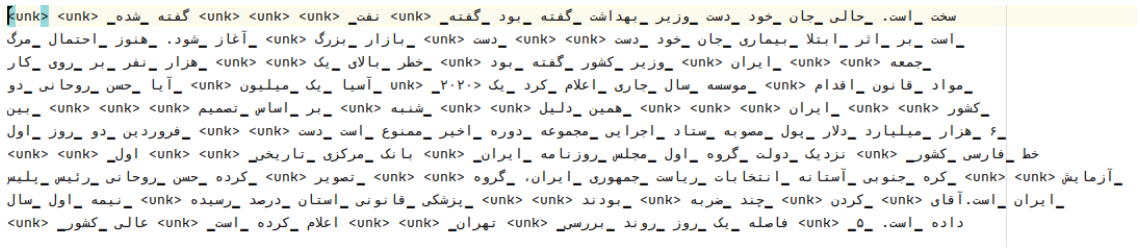

Figure 3.9: Generated Text Transformer language model, label = Iran

in "models/lm" directory. Then run "bash run.sh 41" in order to generate text for each label. For each model generated text will save in corresponding file in "reports/lm" directory.

3.4 Temperature

Different temperature can be used when generating text. If set temperature low, generated text will be more general, as a result number of <unk> token will increase. On the other hand by setting higher value for temperature model diversity increases. For this project 1 is best choice for temperature parameter.

3.5 Final Outputs

In this section output of each label on trained transformer-based language model is represented.

3.5.1 Iran

Iran news is largest corpus in this dataset. We can see more meaningful sentences and less <unk> tokens. Generated sentences seems link Iran news and generated-words illustrated in Figure 3.9 are related to the context.

3.5.2 Art

Longest n-gram in generated text without any <unk> token is placed on second line, which n is equal to 10(Figure 3.10). This sequence of words is not meaningful enough, but seems fluent.

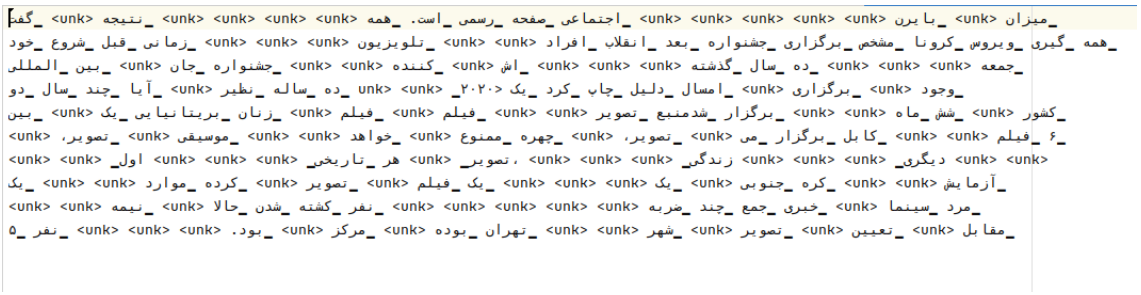

Figure 3.10: Generated Text Transformer language model, label = Art

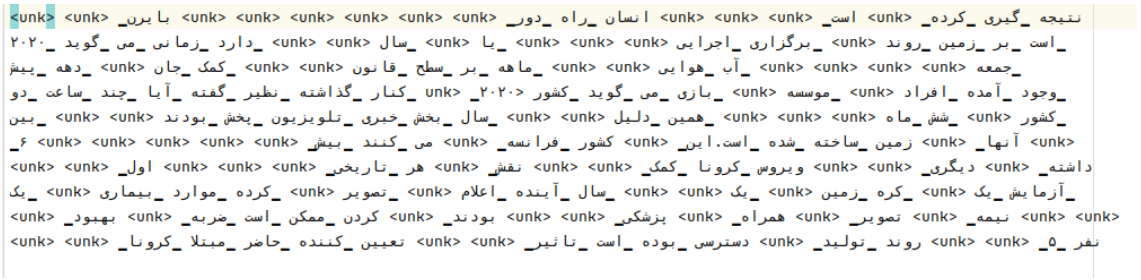

Figure 3.11: Generated Text Transformer language model, label = Science

3.5.3 Science

In this section there are many <unk> tokens due lack of data in Science class. Longest sequence of words is 4-grams and used words are more general than other sections.

3.5.4 Economic

In Economic class generated field as it is shown in figure 3.12 most at least 6-grams are construct of meaningful sequences of words and they sounds same as economic news. We can see many 8-grams without any <unk> token in this section.

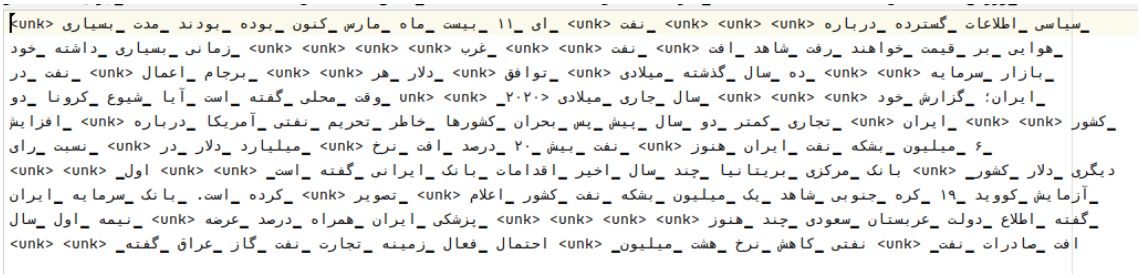

Figure 3.12: Generated Text Transformer language model, label = Economic

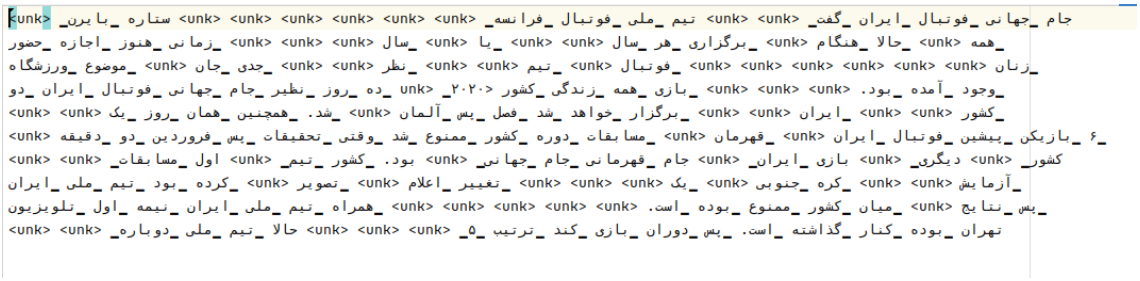

Figure 3.13: Generated Text Transformer language model, label = Sport

3.5.5 Sport

In this class, generated text includes of variety of meaningful n-grams without any <unk> token. It is interesting that fluent and meaning 9-garms sequence of words is constructed.(Line 4, Figure 3.13)