Mathematical Problems of Computer Science 57, 30–38, 2022.
doi: 10.51408/1963-0084

UDC 004.8

Compact N-gram Language Models for Armenian

Davit S. Karamyan[1] and Tigran S. Karamyan[2]

[1]Russian-Armenian University, Yerevan, Armenia
[2]Yerevan State University, Yerevan, Armenia
e-mail: davitkar98@gmail.com, t.qaramyan@ysu.am

Abstract

Applications such as speech recognition and machine translation use language models to select the most likely translation among many hypotheses. For on-device applications, inference time and model size are just as important as performance. In this work, we explored the fastest family of language models: the N-gram models for the Armenian language. In addition, we researched the impact of pruning and quantization methods on model size reduction. Finally, we used Bye Pair Encoding to build a subword language model. As a result, we obtained a compact (100 MB) subword language model trained on massive Armenian corpora.

Keywords: Armenian language, N-gram Language Model, Subword Language Model, Pruning, Quantization.
Article info: Received 31 March 2022; accepted 17 May 2022.

1. Introduction

Language modeling is a fundamental task of NLP. Models that assign probabilities to sequences of tokens are called language models or LMs. Here, tokens can be words, characters, or subwords. The N-gram is the simplest model that assigns probabilities to sentences and sequences of tokens. Although the N-gram models are much simpler than modern neural language models based on RNN[1, 2] and transformers[3, 4, 5], they are much faster than others since they perform constant-time lookups and scalar multiplications (instead of matrix multiplications in neural models). As always, trade-offs exist between time, space, and accuracy[6]. Hence, much recent work has focused on building faster and smaller N-gram language models[7, 8, 9].

N-gram language models are widely utilized in spelling correction[10], speech recognition[11] and machine translation[12] systems. In such systems, for each utterance/sentence translation, the system generates several alternative token sequences and scores them using N-gram LM to peek the most likely translation sequence. In addition, LM rescoring can be combined with beam search algorithms[13].

The Armenian language has a rich morphology: one word can have several tenses and surface forms. Moreover, one can form long words in Armenian by stringing word pieces together. The inclusion of every form in the vocabulary will make it intractable. Subword dictionaries, in which words are divided into frequent parts, can help reduce vocabulary size. Many efforts have been made to use word decomposition and subword LMs for dealing with out-of-vocabulary words in inflective languages such as Arabic[14], Finnish[15], Russian[16], and Turkish[17]. A review of the literature revealed that there have been no publicly available LM resources for the Armenian language. This work is devoted to the creation of a compact and fast N-gram LM for the Armenian language.

Summing up, we will give answers to the following practical questions: **Q1**. What order of N-grams is enough to build a good LM for the Armenian language? **Q2**. How much data is needed to build a model? **Q3**. How can pruning and quantization help reduce the size of the model? **Q4**. Can we build more compact models by using subwords?

In addition, we are going to release training codes and models.[1]

2. Background

Language Modeling (LM) is the task of predicting which token or word comes next. You might also think of an LM as a system that assigns probability to a piece of text. The probability of a sequence of n tokens $t_1^n\{t_1, ..., t_n\}$ is denoted as $P(t_1^n)$. Using the chain rule of probability we can decompose this probability:

$$P(\{t_1, ..., t_n\}) = \prod_{k=1}^{n} P(t_k|t_1^{k-1}).$$

Instead of computing the probability of a token given its entire history, it is usually conditioned on a window of N previous tokens. The assumption that the probability of a token depends only on the previous $N - 1$ token is called a Markov assumption:

$$P(t_k|t_1^{k-1}) \approx P(t_k|t_{k-N+1}^{k-1}).$$

We can estimate the probabilities of an N-gram model by getting counts from a corpus and normalizing the counts so that they lie between 0 and 1. For example, to compute a particular N-gram probability of a token t_k given the previous tokens t_{k-N+1}^{k-1}, we'll compute the count of the N-gram t_{k-N+1}^{k} and normalize it by the sum of all the N-grams that share the same prefix t_{k-N+1}^{k-1}:

$$P(t_k|t_{k-N+1}^{k-1}) = \frac{Count(t_{k-N+1}^{k})}{\sum_t Count(t_{k-N+1}^{k-1}, t)} = \frac{Count(t_{k-N+1}^{k})}{Count(t_{k-N+1}^{k-1})}.$$

There are two major problems with N-gram language models: storage and sparsity. To compute N-gram probability we need to store counts for all N-grams in the corpus. As N increases or the corpus size increases, the model size increases as well. Pruning and Quantization may provide a partial solution to reduce the model size. Any N-gram that appeared a sufficient number of times might have a reasonable estimate for its probability. Since any corpus is limited, some perfectly acceptable tokens may never appear in the corpus. As a result of it, for any training corpus, there will be a substantial number of cases of putative zero probability N-grams. To keep an LM from assigning zero probability to these unseen events, we will have to shave off a bit of probability mass from some more frequent events and give it to the events we have never seen. This is called smoothing. There are many ways to do smoothing: add-one(add-k) smoothing, backoff, and Kneser-Ney smoothing[18].

[1]https://github.com/naymaraq/arm-n-gram

3. Experiments

Setup. We estimate N-gram probabilities on Armenian Wikipedia corpus[2] and CC-100 Web Crawl Data[3][19]. To test the language models, we compute perplexity on two test datasets: Armenian Paraphrase Detection Corpus[4] (ARPA[20]) and Universal Dependencies treebank[5] (UD). All datasets are normalized by removing punctuation marks and non-Armenian symbols. Table 1 provides some statistics of the data after all normalization steps have been performed. Table 2 shows unique N-gram counts presented in the training corpus.

We are going to measure the perplexity of corpus C that contains m sentences and N tokens. Let's the sentences $(s_1, s_2, ..., s_m)$ be part of C. Under assumption that those sentences are independent, the perplexity of the corpus is given by:

$$Perp(C) = \sqrt[N]{\frac{1}{p(s_1, s_2, ..., s_m)}} = \sqrt[N]{\frac{1}{\prod_{k=1}^{m} p(s_k)}}.$$

We use KenLM [21] to train language models. KenLM implements two data structures: Probing and Trie, for efficient language model queries, reducing both time and memory costs. KenLM estimates language model parameters from text using modified Kneser-Ney smoothing.

Table 1: Datasets statistics.

Dataset	Tokens (M)	Bytes	Split
CC-100	409	5.4Gb	train
Wiki	18.6	249Mb	train
ARPA	0.133	1.8Mb	test
UD	0.034	425Kb	test

Table 2: N-gram counts.

Order (N)	Count of unique N-grams
1	3648574
2	60190581
3	160796455
4	217396323
5	233510708

Q1. Order of Grams vs Perplexity

To determine what order of N-grams is sufficient to build a good LM for Armenian, we trained several LMs with different orders and calculated perplexity on the test datasets. Fig. 1 shows the trend between perplexity and order of N-gram. It also shows how the size of the model changes as N increases.

From Fig. 1 we can deduce that the effective orders are 5 and 6 grams. Although their sizes are quite large: 3.9GB and 5.5GB.

Q2. Training Corpus size vs Perplexity

The next question we would like to ask is about corpus size. If the training corpus is small, we will end up with a very sparse model, and all perfectly acceptable Armenian tokens will

[2]https://github.com/YerevaNN/word2vec-armenian-wiki
[3]https://data.statmt.org/cc-100/
[4]https://github.com/ivannikov-lab/arpa-paraphrase-corpus
[5]https://github.com/UniversalDependencies/UD_Armenian-ArmTDP

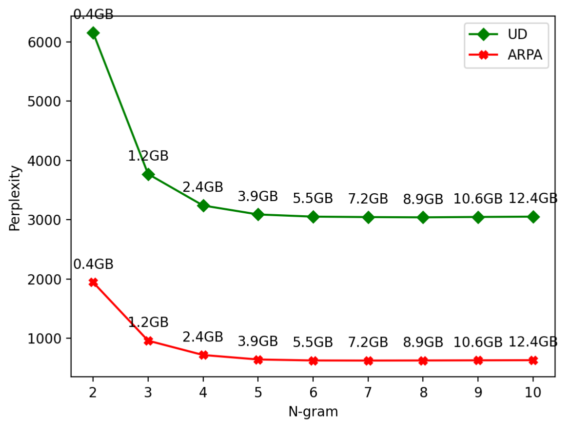

Fig. 1. *N*-gram order vs perplexity.

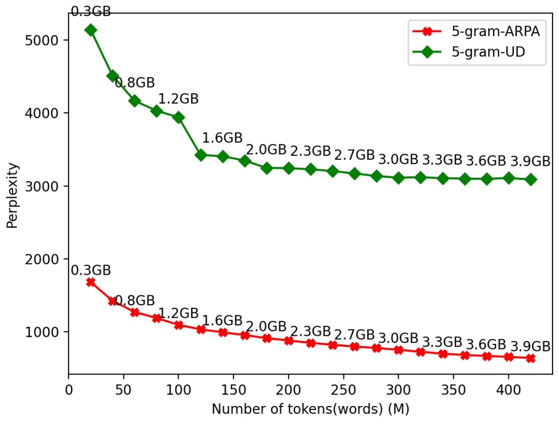

Fig. 2. Number of tokens in training corpus vs perplexity.

be considered unknown. To find out how much data is required, we shuffled and divided the entire training corpus into parts and trained a 5-gram LM for each part. Fig. 2 shows the trend between perplexity and corpus size.

It can be seen that the perplexity reaches saturation when the number of tokens exceeds 380M. Of course, there is always a trade-off between the corpus size, perplexity and the model size: the larger the corpus size, the less perplexity and the larger the model.

Q3. Quantization and Pruning

On-device applications should be as compact as possible. So, the next question we would like to raise concerns the size of the model. Can we build a smaller LM without sacrificing performance?

To reduce the size of the model, we prune all n-grams that appear in the training corpus less than or equal to a given threshold. In addition, we use quantized probabilities by setting fewer bits. For this experiment, we trained a 5-gram LM.

The effect of pruning and quantization is provided in Table 3. Quantization can help reduce the size of a model by a couple of megabytes without perplexity degradation. In contrast, pruning drastically reduces the size of the model at the cost of worsening perplexity. For example, removing all n-grams less than or equal to 4 can reduce the size of the model by more than 12 times with a relative perplexity degradation of 36% for the UD dataset and 100% for the ARPA dataset.

Q4. Subword Language Modeling

So far, we have considered text as a sequence of words separated by a space. Space tokenization is an example of word tokenization, which is defined as breaking sentences into words. The word tokenization method can lead to problems for massive text corpora and usually generates a very big vocabulary (e.g., our training corpus contains $3,648,574$ unique tokens, see Table 1). Instead of using word tokenization, we will use subword tokenization, which is based on the principle that frequently used words should not be split into smaller subwords, but rare words should be decomposed into meaningful subwords. There are several subword

Table 3: The effect of pruning and quantization on the trade-off between size and perplexity.

Pruning threshold	N-bits	Size	UD	ARPA
0	5	3.44Gb	3043.47	631.58
0	6	3.59Gb	3068.62	638.84
0	7	3.74Gb	3075.99	641.57
0	8	3.9Gb	3089.41	642.93
2	5	481.28Mb	3781.29	1131.82
2	6	501.76Mb	3768.36	1128.14
2	7	512.0Mb	3767.81	1125.54
2	8	532.48Mb	3764.69	1125.0
4	5	296.96Mb	4252.71	1344.56
4	6	307.2Mb	4219.03	1335.89
4	7	317.44Mb	4218.13	1332.73
4	8	317.44Mb	4217.73	1332.84
6	5	245.76Mb	4473.19	1486.95
6	6	245.76Mb	4432.03	1474.89
6	7	256.0Mb	4435.29	1471.75
6	8	256.0Mb	4431.23	1471.89
8	5	215.04Mb	4694.73	1588.09
8	6	225.28Mb	4655.29	1576.21
8	7	225.28Mb	4652.95	1571.46
8	8	225.28Mb	4652.89	1571.94

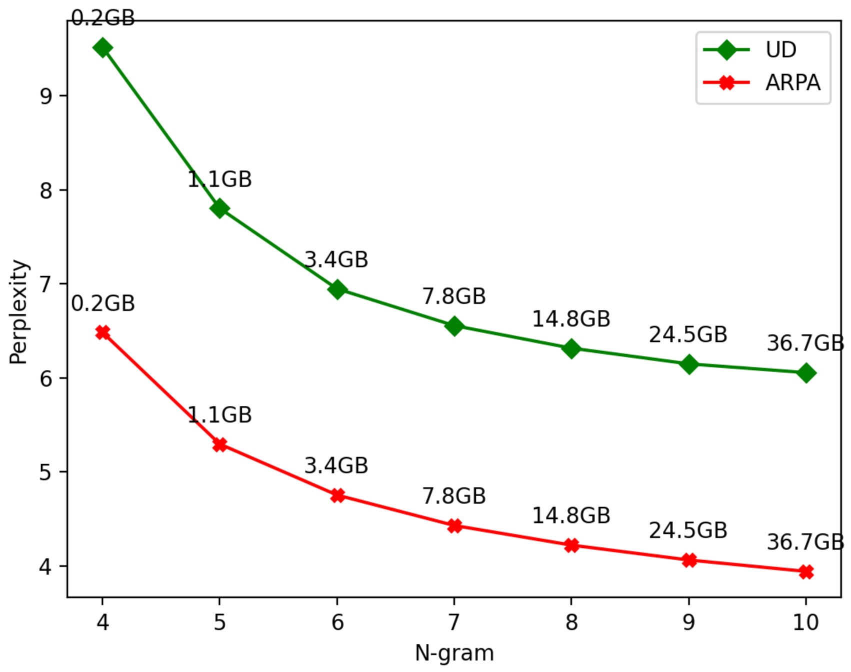

Fig. 3. N-gram order vs perplexity (subword).

tokenization algorithms: Byte-Pair Encoding[22] , WordPiece[23], and SentencePiece[24]. Subword tokenization allows the model to have a reasonable vocabulary size. In addition, subword tokenization enables the model to process words it has never seen before by decomposing them into known subwords. This is especially useful in agglutinative languages such as Armenian, where you can form long words by stringing subwords together.

We trained a BPE tokenizer with a vocabulary size of 128 using the SentencePiece package[6]. Next, we build several N-gram models on a tokenized corpus. Fig. 3 shows the trend between perplexity and order of N-gram for subword model. It also shows how the size of the model changes as N increases.

Table 4: Pruning effect for the subword model with 10-gram.

Pruning	Size	UD	ARPA
0	36.66Gb	6.055	3.941
2	1.11Gb	6.199	4.19
4	634.88Mb	6.323	4.306
6	440.32Mb	6.373	4.381
8	348.16Mb	6.435	4.44
10	286.72Mb	6.53	4.491
16	184.32Mb	6.781	4.619
20	153.6Mb	6.892	4.69
24	122.88Mb	7.02	4.751
30	102.4Mb	7.146	4.837

First, in Fig. 3 the perplexity (0-10) is significantly lower than the perplexity of the word-based tokenized model (0-7000, see Fig. 1). This is because we no longer have unknown tokens. In contrast to word-based models, subword models are much larger (e.g., 10-gram subword model is 3 times bigger).

Since the sequences no longer contain words, but contain subwords, in order to capture sufficient context, we need to consider higher order grams. From Fig. 3 it can be seen that the higher the order, the larger the model (for example, a subword model with 10-gram has a size of 36.7 GB). To reduce the size of the model, we use pruning again. Table 4 provides information about the pruning effect for the subword model with 10-gram. It can be seen that we can reduce the model size by a factor of 368 from 36.7 GB to 102 MB with a relative perplexity degradation of 18% for the UD dataset and 23% for the ARPA dataset.

4. Conclusions

In this article, we have explored N-gram language models for the Armenian language. Our experiments have shown that for word-based language models, the effective orders are 5 and 6. In contrast, the effective order for subword language models can be higher than 10.

We have also explored the impact of pruning and quantization on the trade-off between model size and perplexity. Quantization can help reduce the size of the model without

[6]https://github.com/google/sentencepiece

degrading perplexity significantly. Pruning, on the other hand, drastically reduces the size of the model at the expense of aggravating perplexity. For the subword language model, the perplexity degradation is much lower than for the word-based language model.

We have released compact N-gram language models built on very large corpora.

References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory. Neural computation", vol. 9, no. 8, pp. 1735-1780, 1997.

[2] J. Sarzyska-Wawer, A. Wawer, A. Pawlak, J. Szymanowska, I. Stefaniak, M. Jarkiewicz and . Okruszek, "Detecting formal thought disorder by deep contextualized word representations", *Psychiatry Research*, vol. 304, pp. 114–135, 2021.

[3] J. Devlin, M. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[4] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P.J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer", *arXiv preprint arXiv:1910.10683*, 2019.

[5] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell and others, "Language models are few-shot learners", *arXiv preprint arXiv:2005.14165*, 2020.

[6] C. Buck, K. Heafield and B.V. Ooyen, "N-gram counts and language models from the common crawl", *In: LREC*, vol. 2, no. 4, 2014.

[7] A. Pauls and D. Klein, "Faster and smaller n-gram language models". *In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 258-267, 2011.

[8] D. Guthrie and M. Hepple, "Storing the web in memory: Space efficient language models with constant time retrieval", *In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 262-272, 2010.

[9] U. Germann, E. Joanis and S. Larkin, "Tightly packed tries: How to fit large models into memory, and make them load fast, too", *In: Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing (SETQA- NLP 2009)*, pp. 31-39, 2009.

[10] S.D. Hernandez and H. Calvo, "Conll 2014 shared task: Grammatical error correction with a syntactic n-gram language model from a big corpora", *In: Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pp. 53-59, 2014.

[11] A.Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates and others, "Deep speech: Scaling up end-to-end speech recognition", *arXiv preprint arXiv:1412.5567*, 2014.

[12] H. Schwenk, D. Dchelotte and J. Gauvain, "Continuous space language models for statistical machine translation", *In: Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pp. 723-730, 2006.

[13] A.Y. Hannun, A.L. Maas, D. Jurafsky and A. Y. Ng, “First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns”, *arXiv preprint arXiv:1408.2873*, 2014.

[14] A.E.D. Mousa, H.J. Kuo, L. Mangu and H. Soltau, “Morpheme-based feature-rich language models using deep neural networks for lvcsr of egyptian arabic”, *In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8435-8439, 2013.

[15] V. Siivola, T. Hirsimki, M. Creutz and M. Kurimo, “Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner”, *In: Proc. Eurospeech*, vol. 3, pp. 2293-2296, 2003.

[16] I. Oparin, “Language models for automatic speech recognition of inflectional languages”, *University of West Bohemia*, 2008.

[17] D. Yuret and E. Bicici, “Modeling morphologically rich languages using split words and unstructured dependencies”, *In: Proceedings of the ACL-IJCNLP 2009 conference short papers*, pp.345–348, 2009.

[18] D. Jurafsky, “Speech and language processing”, *Pearson Education India*, 2000.

[19] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmn, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, “ Unsupervised cross-lingual representation learning at scale”, *arXiv preprint arXiv:1911.02116*, 2019.

[20] A. Malajyan, K. Avetisyan and T. Ghukasyan, “Arpa: Armenian paraphrase detection corpus and models”, *In: 2020 Ivannikov Memorial Workshop (IVMEM)*, pp. 35-39, 2020.

[21] K. Heafield, “Kenlm: Faster and smaller language model queries”, *In: Proceedings of the sixth workshop on statistical machine translation*, pp. 187-197, 2011.

[22] R. Sennrich, B. Haddow and A. Birch, “Neural machine translation of rare words with subword units”, *arXiv preprint arXiv:1508.07909*, 2015.

[23] M. Schuster and K. Nakajima, “Japanese and korean voice search”, *In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* pp. 5149-5152, 2012.

[24] T. Kudo and J. Richardson, “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing”, *arXiv preprint arXiv:1808.06226*, 2018.

Կոմպակտ N-գրամ լեզվի մոդելներ հայերենի համար

Դավիթ Ս. Քարամյան[1] և Տիգրան Ս. Քարամյան[2]

[1] Ռուս-հայկական համալսարան, Երևան, Հայաստան
[2] Երևանի պետական համալսարան, Երևան, Հայաստան
e-mail: davitkar98@gmail.com, t.qaramyan@ysu.am

Ամփոփում

Այնպիսի հավելվածներ, ինչպիսիք են խոսքի ճանաչումը և մեքենայական թարգմանությունը, օգտագործում են լեզվի մոդելներ բազմաթիվ վարկածների մեջ ամենահավանական թարգմանությունն ընտրելու համար: Սարքերի վրա տեղադրված հավելվածների համար եզրակացության ժամանակը և մոդելի չափը նույնքան կարևոր են, որքան արտադրողականությունը: Այս աշխատանքում մենք ուսումնասիրել ենք լեզվական մոդելների ամենաարագ ընտանիքը՝ N-gram մոդելները հայերենի համար: Բացի այդ, մենք ուսումնասիրել ենք կտրման և քվանտացման մեթոդների ազդեցությունը մոդելի չափի կրճատման վրա: Ի վերջո, մենք օգտագործել ենք Bye Pair Encoding՝ ենթաբառերի լեզվի մոդել ստեղծելու համար: Արդյունքում ստացել ենք կոմպակտ (100 ՄԲ) ենթաբառերի լեզվի մոդել՝ պատրաստված հայկական զանգվածային կորպուսների վրա:

Բանալի բառեր՝ hայng լեզու, N-gram լեզվի մոդել, ենթաբառերի լեզվի մոդել, էտում, քվանտացում:

Компактные языковые модели N-грамм для армянского языка

Давид С. Карамян[1] и Тигран С. Карамян[2]

[1] Российско-Армянский университет, Ереван, Армения
[2]Ереванский государственный университет, Ереван, Армения
e-mail: davitkar98@gmail.com, t.qaramyan@ysu.am

Аннотация

Такие приложения, как распознавание речи и машинный перевод, используют языковые модели для выбора наиболее вероятного перевода среди множества гипотез. Для приложений на устройстве время вывода и размер модели так же важны, как и производительность. В этой работе мы исследовали самое быстрое семейство языковых моделей: модели N-грамм для армянского языка. Кроме того, мы исследовали влияние методов обрезки и квантования на уменьшение размера модели. Наконец, мы использовали Bye Pair Encoding для построения модели языка подслов. В результате мы получили компактную (100 МБ) модель языка подслов, обученную на массивных армянских корпусах.

Ключевые слова: Армянский язык, модель языка N-грамм, модель языка подслов, обрезка, квантование.