Character-based Language Model

Vít Baisa

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
xbaisa@fi.muni.cz

Abstract. Language modelling and also other natural language processing tasks are usually based on words. I present here a more general yet simpler approach to language modelling using much smaller units of text data: character-based language model (CBLM).[1] In this paper I describe the underlying data structure of the model, evaluate the model using standard measures (entropy, perplexity). As a proof-of-concept and an extrinsic evaluation I present also a random sentence generator based on this model.

Keywords: language model, suffix array, LCP, trie, character-based, random text generator, corpus

1 Introduction

Current approaches to language modelling are based almost utterly on words. To work with words, the input data needs to be tokenized which might be quite tricky for some languages. The tokenization might cause errors which are propagated to following processing steps. But even if the tokenization was 100% reliable, another problem emerges: word-based language models treat similar words as completely unrelated. Consider two words *platypus* and *platypuses*. The former is contained in the latter yet they will be treated completely independently. This issue can be sorted out partially by using factored language models [1] where lemmas and morphological information (here singular vs. plural number of the same lemma) are treated simultaneously with the word forms.

In most systems, word-based language models are based on n-grams (usually 3–4) and on Markov chain of the corresponding order where only a finite and fixed number of previous words is taken into account. I propose a model which tackles with the above-mentioned problems. The tokenization is removed from the process of building the model since the model uses sequences of characters (or bytes) from the input data. Words (byte sequences) which share prefix of characters (bytes) are stored on the same place in the

[1] I call this ChaRactEr-BasEd LangUage Model (CBLM) *cerebellum*: a part of human brain which plays an important role in motor control and which is involved also in some cognitive processes including language processing.

model. The model uses suffix array and trie structure and is completely language independent.

The aim of CBLM is to make language modelling more robust and at the same time simpler: with no need for interpolating, smoothing and other language modelling techniques.

2 Related work

Character-based language models are used very rarely despite they are described frequently in theoretical literature. That is because a standard n-gram character-based language models would suffer from very limited context: even 10-grams are not expressive enough since they describe only very limited width of context. Even the famous Shannon's paper [2] mentions a simple uni-, bi- and tri-gram models but then it swiftly moves to word-based models.

There have been some attempts to use sub-word units (morphemes) for language modelling, especially for speech recognition tasks [3] for morphologically rich languages like Finnish and Hungarian but they have not gone deeper.

Variable-length n-gram modelling is also closely related but the model described in [4] is based rather on categories than on substrings from the raw input data. Suffix array language model (SALM) based on words has been proposed in [5].

3 Building the model

As input, any plain text (in any encoding but it is most convenient to use UTF-8) can be used. In a previous version of the model all input characters were encoded to a 7-bit code (the last bit was used for storing structure information). Currently the model requires a simpler preprocessing: the data is taken as is—as a sequence of bytes. Sentences are separated by a newline character. The only pre-processing is lower-casing—quite common practice.

3.1 Suffix array, longest common prefix array

The next step is suffix array construction. Suffix array (SA) is a list of indexes (positions) in the input data which are sorted according to lexicographical order of suffixes starting at the corresponding positions in the data. The Table 1 shows an example of a SA constructed for string *popocatepetl*. The resulting SA is in the third column. The last column contains longest common prefix array (LCP) which corresponds to a number of common characters between two consecutive suffixes in the SA.

I use *libdivsufsort*[2] library for fast SA construction in $O(n \log n)$ time where n is input data size in bytes. The size of an input is limited to 2 GB since longer

[2] `https://code.google.com/p/libdivsufsort/`

Table 1: Suffix array example

I	suffix	SA	sorted suffix	LCP
0	`popocatepetl`	5	`atepetl`	0
1	`opocatepetl`	4	`catepetl`	0
2	`pocatepetl`	7	`epetl`	0
3	`ocatepetl`	9	`etl`	1
4	`catepetl`	11	`l`	0
5	`atepetl`	3	`ocatepetl`	0
6	`tepetl`	1	`opocatepetl`	1
7	`epetl`	8	`petl`	0
8	`petl`	2	`pocatepetl`	1
9	`etl`	0	`popocatepetl`	2
10	`tl`	6	`tepetl`	0
11	`l`	10	`tl`	1

data could not be encoded using 4-byte integer indexes. The size of a compiled SA is $O(n \log n)$.

LCP is computed separately using an inverse SA in $O(n)$ time and $O(n)$ space. To limit the size of LCP array, the highest possible number in LCP array is 256 (1 B per item). Thus the longest substring which can be stored in the model has length 256 characters (bytes).

3.2 Trie

Once SA and LCP are built, all prefixes from SA which occur more than $N\times$ are put into trie structure. The N is the only parameter used in construction of the trie. Each node in the trie stores probability (relative frequency) of occurrence of the corresponding prefix in SA.

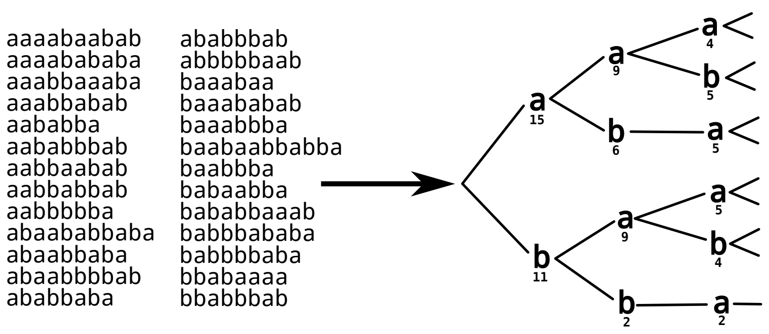

Fig. 1: Construction of a trie from an example suffix array

In Figure 1 you can see an example of suffix array turned into trie structure. Only the upper part of the trie is shown. The numbers below the nodes correspond to frequencies of the prefixes.

The trie is stored in a linear list—the tree structure of the trie is preserved using integer indexes of the list. Each item of the list stores 4 slots: 1) an address of the eldest children, 2) probability of the current prefix, 3) the character byte itself and 4) binary true or false: if the current item is the last node in a row of siblings.

The siblings are sorted according probability. In Figure 2 there is an example for substring *barb* from a Czech model. It is obvious that after the prefix, characters *a, o, i* and *e* are the most frequent. They occur in words like *barbar, barbora, barbie, barbecue, rebarbora* etc. The dots in the Figure mean a space skipped between the trie items (nodes). After substring *barbo*, the most probable characters are *r* (75%) and in *ř* (22%). See the last two items in Figure 2. Character *ř* is in fact represented by two nodes: a node with byte value 197 and its children node (153) but here I have simplified it.

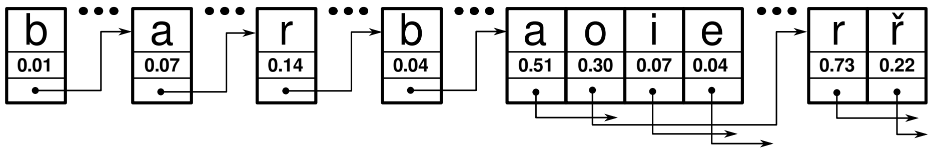

Fig. 2: Trie structure

4 Language model

A language model is a prescription for assigning probabilities to an input data (in this case a sequence of bytes). Here I present a straightforward algorithm using the trie as an underlying data structure. It is important to emphasize that this approach is only a first attempt at language modelling using the trie described above. Further improvements are to be implemented.

Each node in a trie contains probability distribution of all following characters (bytes). The longer is the path from root to a node, the more accurate is the probability distribution (and also the lower entropy and perplexity) in the node since a longer path means a longer context. Following is description of the algorithm: how to compute a probability of any sequence of bytes.

The algorithm starts from the first byte in the data and from the root of the trie. If the byte is among the root's children, the initial probability 1.0 is multiplied by the corresponding probability stored in the relevant child node. The algorithm stores the position of the child node and repeats the procedure for next bytes in the input data. If a current byte is not found among children at a current position, the current prefix (defined by a path from the root to the current position) is shortened from the left (by one byte) and the shortened path is translated to the root (the same character bytes but a different path). It holds that if a path (sequence of bytes) is found wherever in the trie then it must be translatable to the root of the trie. It is a property of the original suffix array. After the translation, the lookup is repeated. If the byte is not found, the path is shortened from the left again and translated again until the byte is found among descendants of the last node in the translated path. It may occur that the path is shortened to an empty path. In that case the procedure continues from the root as at the beginning of the algorithm. Every time a byte from the input data is found among a children of a current position, the overall probability is multiplied by the probability of the children.

It is necessary that probability of any subsequence of the input data is greater than 0 otherwise the result probability would be zero too. In n-gram models the solution is achieved by smoothing language models using techniques designed like Katz, Kneser-Ney or Good-Turing smoothing. In CBLM, the problem of zero probability (caused by symbols which are in the input data but do not occur in an input data more than $N\times$) is solved by assigning probability p_r to all unseen symbols. Probability p_r is taken from the first level in trie—the complement of the sum of probabilities of all child nodes of the root. For one English model trained on Orwell's *1984* $p_r = 0.000018$ since some symbols (+, %) occurred less than $N\times$.

The described procedure above can be slightly modified to obtain a random text generator (see Section 6). The bytes are not read from the input but generated randomly from distribution probabilities stored in nodes.

5 Intrinsic evaluation: entropy per byte

The standard way to evaluate language models is to measure entropy per word. In the case of this model I use entropy H and perplexity PP per byte. The formulas for test data $b_1 \cdots b_N$ and model m are as follow:

$$H(m) = -\frac{1}{N}\sum_{i=1}^{N} \log p(b_i)$$

$$PP = 2^{H(m)}$$

where N is length of the test data and $p(b_i)$ is probability given by the algorithm.

Table 2 shows results for English and Czech data. The models for English has been built from British National Corpus and from George Orwell's *1984*. The test data were Lewis Carroll's *Alice in Wonderland* and George Orwell's *1984*. The models for Czech has been built from the Czech National Corpus, Czech Wikipedia corpus (csWiki) and Czech Web corpus czTenTen[3] [6] (csWeb).

[3] `http://www.sketchengine.co.uk/documentation/wiki/Corpora/czTenTen2`

Table 2: Evaluation of Czech and English models on Lewis Carroll's *Alice in Wonderland* and George Orwell's *1984*.

Model	Test	Size	H	PP
BNC_2	Alice	330 M	2.286	4.879
BNC_3	Alice	212 M	2.294	4.904
BNC_2	1984	330 M	**1.664**	3.170
BNC_3	1984	212 M	1.671	3.184
$SYN2000_4$	1984	362 M	1.837	3.574
$csWiki_5$	1984	300 M	1.850	3.607
$csWeb_4$	1984	312 M	**1.571**	2.972

Some observations from the tables follow. The BNC_2 model has achieved only a slightly better entropy and perplexity than BNC_3 for both test data. Notable is also the fact that both models assign considerably higher entropy and perplexity to *Alice*. It is probably caused by the peculiar language of Carroll. For comparison—the entropy of English has been estimated to 1.75 [7]. The best Czech model is $csWeb_5$ which obtained 1.571 entropy for Orwell's *1984*.

The performance of the Czech and English models is quite stable. When Markov model N parameter is fixed, performance (perplexity) differs substantially when languages from different language families are evaluated. See also the comparable performance of Hungarian and English random generator in Section 6. Further intrinsic evaluation is to be carried out using a standard statistical language modelling benchmark, e.g. one billion word benchmark [8] or Brown corpus.

6 Extrinsic evaluation: random sentence generator

It has been reported that a language model perplexity measure is not correlating well with the evaluations of applications in which the model is used (e.g. [9]). That is why some researchers prefer extrinsic evaluation methods over intrinsic measures. To provide an extrinsic evaluation of the presented model I have developed a simple random text generator based on CBLM.[4] It offers models for English, Georgian, Hungarian, Japanese, Russian, Turkish, Slovak, Czech and Latin. Users may save generated sentences (*Like* link) which are then available as favourite sentences at the bottom of the web page.

The algorithm is a modification of the language model algorithm. It generates a random stream of bytes and whenever it generates a second new line character, the result is written to the output. By using the sequence of bytes between two newlines, the generator is capable of generating texts roughly on sentence level.

[4] `http://corpora.fi.muni.cz/cblm/generate.cgi`

To increase legibility, initial letters and named entities in the following example sentences have been upper-cased. The source data for the examples were as follow. English: British National Corpus, Czech: czTenTen (first 600 M tokens), Hungarian: Hungarian Wikipedia, Latin: a Latin corpus and Slovak: Slovak Wikipedia. In almost all cases $N = 3$.

English First there is the fact that he was listening to the sound of the shot and killed in the end a precise answer to the control of the common ancestor of the modern city of Katherine street, and when the final result may be the structure of conservative politics; and they were standing in the corner of the room.

Czech Pornoherečka Sharon Stone se nachází v blízkosti lesa. ¶ Máme malý byt, tak jsem tu zase. ¶ Změna je život a tak by nás nevolili. ¶ Petrovi se to začalo projevovat na veřejnosti. ¶ Vojáci byli po zásluze odměněni pohledem na tvorbu mléka. ¶ Graf znázorňuje utrpení Kristovo, jež mělo splňovat následující kritéria.

Hungarian Az egyesület székhelye: 100 m-es uszonyos gyorsúszásban a következő években is részt vettek a díjat az égre nézve szójaszármazékot. ¶ Az oldal az első lépés a tengeri akvarisztikával foglalkozó szakemberek számára is ideális szállás költsége a vevőt terhelik.

Slovak Jeho dizajn je v zrelom veku, a to najmä v prípade nutnosti starala sa o pomoc na rozdiel od mesta prechádza hranica grófstva Cork. ¶ V roku 2001 sa začala viac zameraný na obdobie 2006 prestúpil do monastiera pri rieke Marica, ktorú objavil W. Herschel 10. septembra 1785.

Latin Quinto autem anno andum civitates et per grin in multitudinem quae uerae e aeque ad omne bonum. ¶ Augustinus: video et oratteantur in caelum clamor eus adiutor meus, et uit, quam hoc crimen enim contentit et a debent.

7 Future work & conclusion

I have described a first approximation of the language model. In the future I want to exploit more some properties of the trie model, e.g. probabilities of sequences which follow a given sequence (not necessarily immediately following it). This would allow to express connections (associations) between any two byte sequences and to capture a broader context.

Acknowledgement This work was partially supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

References

1. Bilmes, J.A., Kirchhoff, K.: Factored language models and generalized parallel backoff. In: Proceedings of the 2003 Conference of the North American Chapter

of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2, Association for Computational Linguistics (2003) 4–6
2. Shannon, C.: A mathematical theory of communication. Bell Sys. Tech. J. **27** 379–423
3. Creutz, M., Hirsimäki, T., Kurimo, M., Puurula, A., Pylkkönen, J., Siivola, V., Varjokallio, M., Arisoy, E., Saraçlar, M., Stolcke, A.: Morph-based speech recognition and modeling of out-of-vocabulary words across languages. ACM Transactions on Speech and Language Processing (TSLP) **5**(1) (2007) 3
4. Niesler, T.R., Woodland, P.: A variable-length category-based n-gram language model. In: Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on. Volume 1., IEEE (1996) 164–167
5. Zhang, Y., Vogel, S.: Suffix array and its applications in empirical natural language processing. Technical report, Technical Report CMU-LTI-06-010, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA (2006)
6. Suchomel, V.: Recent czech web corpora. In: 6th Workshop on Recent Advances in Slavonic Natural Language Processing. Brno
7. Brown, P.F., Pietra, V.J.D., Mercer, R.L., Pietra, S.A.D., Lai, J.C.: An estimate of an upper bound for the entropy of english. Comput. Linguist. **18**(1) (March 1992) 31–40
8. Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P.: One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. arXiv preprint arXiv:1312.3005. 2013.
9. Iyer, R., Ostendorf, M. Meteer, M.: Analyzing and predicting language model improvements. Automatic Speech Recognition and Understanding, 1997. 254–261