A System for Predictive Writing

Zuzana Nevěřilová and Barbora Ulipová

Computational Linguistics Centre
Faculty of Arts, Masaryk University
Arne Nováka 1, 602 00 Brno, Czech Republic
xpopelk@fi.muni.cz, b.ulipova@gmail.com

Abstract. Most predictive writing systems are based on n-gram model with different size. Systems designed for English are easier than those for flective languages since even smaller models allow reasonable coverage. However, the same corpus size is significantly insufficient for languages with many word forms. The paper presents a new predictive writing system based on n-grams calculated from a large corpus.
We designed the high-performance server-side script that returns either the most probable endings of a word or the most probable following words. We also designed the client-side script that is suitable for desktop computers without touchscreens.
We calculated 150 millions most frequent n-grams for $n = 1, \ldots, 12$ from a Czech corpus and evaluated the writing system on Czech texts. The system was then extended by custom-built model that can consist of domain or user specific n-grams. We measured the key stroke per character (KSPC) rate in two different modes: one – called *letter KSPC* – excludes the control keys since they are input method specific, the other – called *real KSPC* – includes all key strokes. We have shown that the system performs well in general (letter KSPC on average was 0.64, real KSPC on average was 0.77) but performs even better on specific domains with the appropriate custom-built model (letter KSPC and real KSPC were on average 0.63 and 0.73 respectively).
The system was tested on Czech, however it can easily be adapted an arbitrary language. Due to its performance, the system is suitable for languages with high inflection.

Keywords: predictive writing, n-gram language model, corpus, KSPC

1 Introduction

Predictive writing is a useful and popular feature embedded in modern web browsers and cell phones where either endings of an unfinished word or a following word are suggested to the user. Predictive writing is used in order to save the number of key strokes and prevent spelling errors. Some users may even use it to find the correct spelling of words they are not sure about. For these reasons, predictive writing applications are not only useful for cell phones and tablets but they can also support writing on conventional keyboards.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 11–18, 2014.

We present a predictive writing web application that is suitable for devices with conventional keyboards. The application is based on the client-server architecture and thus it can benefit from server perfomance: the language model can be much larger (and thus more precise) than language models that have to use the limited memory space of mobile devices.

This paper is organized as follows: Section 2 overviews in short the related work, in Section 3, we present our system and its extension with custom-built model. In Section 4, we describe the settings of the evaluation experiment and the experiment itself. Section 5 discusses the results. In Section 6, we propose further work for the future.

2 Related Work

Prediction is well established in many tasks e.g. in web browsers and search engines: a web browser usually keeps the search history, and search engines use data obtained from their users as well, e. g. Google uses a combination of phrases from search history of a particular user and overall search history. In addition, it adds information from Google+ profiles[1]. The aim of this prediction is to check spelling and reduction of number of characters the user has to type.

Historically, predictive writing became popular on cell phones with numeric keyboard. Users became accustomed to cluster keyboards (i.e. keyboards with highly ambiguous keys) such as Tegic T9 or Motorola iTap. For example, [1] used vocabularies of 10k, 40k, 160k and 463k words (i.e. unigrams) together with up to 5 milion n-grams (for $n = 2, 3$). The authors concluded that n-gram models are more robust than unigram models (such as T9).

Predictive keyboards on today's mobile devices serve a slightly different purpose: the typing error rate on software keyboards is higher than on hardware keyboards. [2] reported "average number of errors on software keyboards 4.55%, and the average number of hardware was 1.36%". It is thus more comfortable when users do not have to type much.

[3] proposes a simple measure to characterize text entry techniques: key strokes per character (KSPC) "is the number of key strokes required, on average, to generate a character of text for a given text entry technique in a given language". The exact calculation of KSPC depends on both hardware (i.e. it is different on touchscreen keyboards, hardware keyboards, or stylus tapping) and software (i.e. how many characters the user has to tap before the desired word is available to select). For this reason, we calculated two times: KSPC excluding the control keys (the arrows) and KSPC including all keys. In this paper, we call the measures *letter KSPC* and *real KSPC* respectively. The letter KSPC includes the tab key since it is the key stroke that leads to selection of a particular word.

Since the cited text input techniques were primarily developed for English, less attention is paid to non-English texts. For example, [4, p. 9] only state

[1] `https://support.google.com/websearch/answer/106230?hl=en`

that entry of characters not present in the English alphabet increases the number of key strokes required. [4] have shown that n-gram based models for statistical prediction are less reliable for inflected languages but "still offer quite reasonable predictive power".

3 The Predictive Writing System

3.1 Server-side design

We implemented a predictive writing server-side script that benefits from n-grams calculated from the czTenTen corpus. This corpus is currently one of the largest corpora for Czech[2]. The data were collected from different websites, cleaned and deduplicated by the corpora tools [5]. Since the data source contains mainly texts that were not proof-read, the n-grams do not necessarily contain only correct Czech. This issue can be serious when using predictive writing for spelling purposes.

We calculated bigrams using the `lscbgr` tool and filtered out all bigrams with minimum sensitivity < 0.0001. We calculated n-grams for $n = 3, \ldots, 12$ using the `lscngr` tool, then we filtered out all n-grams with frequency of the respective (n-1)-gram representing the n-gram without its last token ($freq_{n-1}$) less than 10. For each such n-gram, we then calculated the score as $n \cdot freq_{n-1}$. in order to prefer longer n-grams. For unigrams, we used the `lsclex` tool[3] and we took all tokens with frequency greater than one and length smaller than 30 characters. All the mentioned parameters were set by experiments. The aim was to generate a database of n-grams (for $n = 1, \ldots, 12$) with a plausible coverage on texts but at the same time with a reasonable size. We always took case-sensitive words, numbers, and punctuation as tokens.

The predictive writing server-side script works basically in two modes:

1. If the input ends with a space, it suggests the following words, i.e. it returns the first 10 most frequent n-grams. The input is truncated to last 12 tokens and compared to the n-gram database. We strongly prefer longer n-grams, so the output is sorted by n-gram size and then by the n-gram score.
2. If the input does not end with a space, it suggests possible endings of the last word. The calculation is based on previous 11 tokens and the unfinished token.

The server-side script functionality is quite simple but for performance reasons, the n-grams were stored in finite state automata (FSA) using the `fsa` package[4].

[2] In Nov 2014, it contained 5,069,447,935 tokens and 4,175,089,440 words, see `https://ske.fi.muni.cz/auth/corpora/`

[3] All tools are documented in the Sketch Engine Project Wiki: `http://www.sketchengine.co.uk/documentation/wiki/SkE/NGrams`.

[4] `http://galaxy.eti.pg.gda.pl/katedry/kiw/pracownicy/Jan.Daciuk/personal/fsa.html`

3.2 Client-side Design

The client side provides a text area for writing and selectbox with suggestions. Unlike touchscreens, the selectbox is controlled by up/down arrows and the tab key for selecting the desired word. We reduced the number of suggestions to 6 since users do not usually find it productive to read more of them. A screentshot is presented in Figure 1. The client side also trims spaces before punctuation.

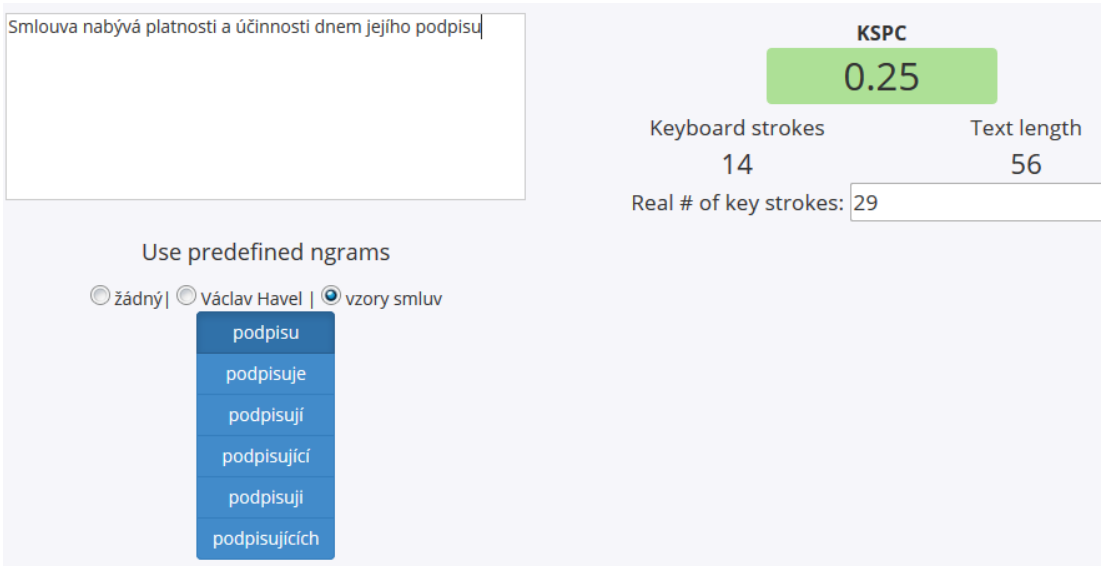

Fig. 1: Screenshot of the basic client

We strongly benefit from asynchronous JavaScript (AJAX) in order to call the server-side script after each key stroke. This procedure is quite demanding on the server-side script performance.

3.3 Extension to custom-built model

Our next goal was to improve the system tailoring it to the individual users or domains. This was done by allowing the user to upload a file with user specific (or domain specific) texts and then adjusting the system so it suggests primarily the n-grams taken from user files: we call these n-grams the *custom-built model*.

3.4 Discussion on the vocabulary size

Czech is a highly inflected language, thus it has much more word forms than e.g. English. The size of the n-gram vocabulary has to be considerably bigger. For example, [6] argue that German corpora have to be 4 times larger than English ones in order to keep the same theoretical minimum error rate in speech recognition. For Hungarian, the corpus size must be 20 times bigger compared to an English corpus.

From the czTenTen corpus, we extracted n-grams for $n = 1, ..., 12$. Table 1 shows the number of n-grams for respective n.

Table 1: Number of n-grams with respect to different n

n	number of n-grams
1	8,312,152
2	68,217,654
3	38,781,821
4	22,163,068
5	8,554,953
6	2,798,732
7	849,500
8	267,789
9	100,753
10	50,674
11	31,152
12	21,140
total	150,149,388

3.5 Building custom language models

To create a file with user n-grams, we first tokenize the user file which is expected to be in a plain text format. We use the `unitok` tool[5]. The corpus data are already tokenized so they do not have to go through this process. To find unigrams and their frequency distribution, we used a bash pipeline:

```
cat inputfilename | /corpora/programy/unitok.py -n | sort \
| uniq -c | sort -r -n | awk '{print $2 ":" $1}' > unigrams
```

This command will simply sum the number of occurrences of each word.

3.6 N-gram weighting

To find longer n-grams, we implemented a function which used the above mentioned `unitok` tool for tokenization and functions from the python nltk toolkit[6] for creating the n-grams and counting the number of their occurrences. Since we do not expect the user data to be very large, we decided to only count n-grams of the length of 2 to 4 tokens. Longer n-grams are not likely to occur more than once in a short text. Also, we did not want their frequency

[5] `https://www.sketchengine.co.uk/documentation/wiki/Website/LanguageResourcesAndTools`

[6] `http://www.nltk.org`

distribution to be the only criteria for sorting because longer n-grams are much less frequent than short ones but at the same time they are much more interesting for our purposes.

We decided to count the n-gram score according to the following formula:

$$Score = FrequencyDistribution \times n^4,$$

where n is the number of tokens in the n-gram. Then we sort the n-grams according to this score.

User n-grams are stored in text files, the server-side script uses the `sgrep` utility[7] for binary search in the text files.

4 Evaluation

Users write the text in a text area and control the prediction via the up/down arrows and the tab key. If they see the desired word in the selection box, they choose it using the arrow keys and then select the word by the tab key. We evaluated the application on several texts measuring both the letter KSPC and the real KSPC (see Section 2). Letter KSPC reflects only the performance of the language model while real KSPC also reflects the input method implementation. Corrections, deletions, and clipboard operations were not comprised in none of the measures.

4.1 Experiment

We measured KPSC on four texts that are not present in the corpus: The average letter KPSC was 0.64 and the real KSPC was 0.77.

Afterwards, we measured the KPSC according to a particular writing style. We used two completely different custom-built models: contract templates and Václav Havel's speeches. From the former source, we obtained 20k unigrams and 386k n-grams for $n = 2, 3, 4$. From the latter source, we obtained 41k unigrams and over milion n-grams for $n = 2, 3, 4$.

We then typed in a paragraph from a mortgage contract (480 characters without spaces), a contract of sale (362 characters) and a part of Václav Havel's speech (524 characters) and a paragraph from Václav Havel's essay (470 characters) with the use of the custom-built models and without them (so the tool is only using data from the general corpus). The paragraphs are long enough so the KSPC is not much influenced by a few out-of-vocabulary words. The results are shown in Table 2.

5 Results and Observations

After uploading the user data, both letter KSPC and real KSPC improved slightly. The measured KPSC shows that the tool is usable for writing arbitrary

[7] `http://sgrep.sourceforge.net/`

Table 2: Resulting KSPC on four texts

text type	mortgage contract	contract of sale	H. speech	H. essay
letter KSPC	0.63	0.55	0.68	0.71
letter KSPC with custom-built model	0.62	0.54	0.67	0.70
real KSPC	0.77	0.64	0.81	0.84
real KSPC with custom-built model	0.70	0.62	0.81	0.77

texts in Czech. Our system is comparable with other prediction systems, e.g. WordTree [7] which reports KSPC = 0.71. Due to the n-gram resource – the web corpus – it can contain non-standard n-grams, thus it is less suitable for spell checking.

The tool is more successful with contracts than Václav Havel's texts. Legal texts have much more predictable sentence structure, many commonly used phrases (resulting in n-grams with higher scores) and a vocabulary where there are lot of words used very often and fewer words are used rarely. This is common with NLP tools which are often more successful with expert domains than general texts.

With Václav Havel's texts the custom-built model improves the result as a whole but sometimes it actually make the result worse. For example, without custom-built model, after typing "já" (the pronoun *I*), the system suggests "bych" (*would*) because the strongest n-gram starting with "já" is "já bych chtěl" (*I would like*). With model built from Havel's texts, the prediction system suggests "se", as Havel's strongest n-gram is "já se domnívám" (*I assume*). However, if we type only "D" without any custom-built model, the system suggests "Dobrý" (*Good*) at the third place while model built from Havel's texts, it does not suggest "Dobrý" at all. Therefore, it will not help when a speech starts with (very common greeting) "Dobrý den" (*Hello* but literally *Good day*).

The program often suggests a word with the right base but a wrong ending. This is due to Czech being a highly inflected language. At the same time, the sentence parts with obligatory agreements do not need to be close.

6 Conclusion and Future Work

We have built a prototypical system for predictive writing and evaluated it on Czech. While predictive writing seems to be the domain of mobile devices, we found several benefits of predictive writing arising from the reduction of number of key strokes: typing speed, correct spelling, natural collocations. These benefits can be arguable and have to be measured on real-world texts. We extended the n-gram model calculated from a general corpus by user specific or domain specific texts. We proved that in some domains, predictive writing with the appropriate custom-built model can be even more effective.

Predictive writing has many applications and potential users such as motor impaired persons, users with dysgraphia, foreigners learning Czech, and touchscreen users. For this reason, we plan to improve the system and to measure its usefulness (expressed by not only the KSPC but also typing speed and number of errors) on real-world texts.

For near future, we plan to clear the n-grams in order to exclude n-grams with undoubtedly incorrect Czech and spelling errors. At the same time, we expect the KSPC decrease when the system offers more than one next word.

Other improvements comprise shift from n-gram to grammar-based prediction and learning from the user input.

Acknowledgements This work has been partly supported by the Masaryk University within the project *Čeština v jednotě synchronie a diachronie – 2014* (MUNI/A/0792/2013).

References

1. Klarlund, N., Riley, M.: Word n-grams for cluster keyboards. In: Proceedings of the 2003 EACL Workshop on Language Modeling for Text Entry Methods. TextEntry '03, Stroudsburg, PA, USA, Association for Computational Linguistics (2003) 51–58
2. Hasegawa, A., Yamazumi, T., Hasegawa, S., Miyao, M.: Evaluating the input of characters using software keyboards in a mobile learning environment: A comparison between software touchpanel devices and hardware keyboards. In: IEEE International Conference on Wireless, Mobile, and Ubiquitous Technology in Education. (2012) 214–217
3. MacKenzie, I.: KSPC (keystrokes per character) as a characteristic of text entry techniques. In Paternò, F., ed.: Human Computer Interaction with Mobile Devices. Volume 2411 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2002) 195–210
4. Matiasek, J., Baroni, M., Trost, H.: FASTY - a multi-lingual approach to text prediction. In Miesenberger, K., Klaus, J., Zagler, W., eds.: Computers Helping People with Special Needs. Volume 2398 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2002) 243–250
5. Suchomel, V., Pomikálek, J.: Efficient web crawling for large text corpora. In Kilgarriff, A., Sharoff, S., eds.: Proceedings of the seventh Web as Corpus Workshop (WAC7), Lyon (2012) 39–43
6. Németh, G., Zainkó, C.: Word unit based multilingual comparative analysis of text corpora. In Dalsgaard, P., Lindberg, B., Benner, H., Tan, Z.H., eds.: INTERSPEECH, ISCA (2001) 2035–2038
7. Badr, G., Raynal, M.: WordTree: Results of a word prediction system presented thanks to a tree. In Stephanidis, C., ed.: Universal Access in Human-Computer Interaction. Applications and Services. Volume 5616 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2009) 463–471