The Impact of Word-Order Constraints on Predictive Sentence Generation

Daniel Bidulock

Dept. of Computer Science, Faculty of Science, University of Calgary, 2500 University Drive N.W., Calgary, Alberta, Canada T2N 1N4
dsbidulo@ucalgary.ca

Abstract. Predictive Sentence Generation is a system analogous to the predictive text systems familiar to cell phone and PDA users. But unlike predictive text, which assists in word completion, Predictive Sentence Generation attempts to assist the user in completing entire sentences. *ParsePS*, a Predictive Sentence Generator, was used to demonstrate the impact contextual considerations have on prediction accuracy. It was determined that word-order constraints do improve accuracy when coupled with an unconstrained prediction mechanism. The constrained mechanism, executed alone, under performs when measured against the baseline set by the unconstrained mechanism.

1 Introduction

Predictive Sentence Generation is a system that attempts to expedite data entry via a keyboard or some similar device. It records and analyzes a user's typing habits over time and *learns* to offer suggestions as to what word might come next in a sentence. Given a corpus, it is possible to do this by simply calculating the probability that one word follows another. The purpose of this research is to determine how Predictive Sentence Generation is impacted when the scope of probabilistic analysis is expanded. That is, in determining the effectiveness of the predictive mechanism, the probability that one word is *preceded* by another will be considered alongside the probability of the word that might follow.

2 Problem Description

The increasing popularity of handheld computing devices with full QWERTY keyboards could potentially eliminate the need for predictive text. However, current hardware can still be somewhat cumbersome to use. For instance, these devices often have tiny buttons or imprecise touch screens. Though the interfaces are improving steadily, user input is still being stymied. Even though a handheld device may have a full keyboard, the user can typically only use thumbs to type, thus severely limiting typing speed.

This research may also have applications in the area of computer accessibility. Predictive sentence generation could help physically disabled users contend with conventional desktop computing devices. Shaky hands, limited motion, chronic pain, and other ailments present an obstacle to efficient keyboard use. Predictive sentence generation may prove to be a viable method with which to augment a disabled user's interaction with a computer.

3 Background

3.1 Web Query Recommendation

Perhaps the most recognizable example of Predictive Sentence Generation is that found in the most popular search engine query entry fields. When a keyword is entered, the search engine presents a dropdown list containing potential search suggestions. Sometimes these suggestions can be quite helpful, while at other times they are of no help at all. It is in offering the helpful suggestions where the major challenges lie[1][2][3]. On the surface, the problem of offering helpful, personalized web query recommendation is related to the problem faced by Predictive Sentence Generation.

Of course, the motivation behind finding solutions to the problem of Predictive Sentence Generation and web query recommendation are quite different. With regard to query assistance, the reviewed solutions all track a user's past behaviour in an attempt to satisfy current search needs. This modeling of past behaviour provides the common thread through web query recommendation and Predictive Sentence Generation. Web query suggestion methods typically focus on *collaborative filtering*[2], *content-based filtering*[3], or analyzing the search patterns of other web users[1]. Collaborative-based filtering depends on users' ratings of a given resource. Content-based filtering depends on a resource's corpus. This final method, involving user search pattern analysis, is of the most interest here.

Like search pattern analysis, Predictive Sentence Generation relies on a dynamic, expanding corpus. The two methods are different, however, in that Predictive Sentence Generation relies on the data entered by only one user. Search pattern analysis, on the other hand, draws from the search terms entered by a multitude of users[1]. Though the source of the data is different, both systems share common characteristics and the same broad goal: analyze a user's past behaviour in order to anticipate current needs, with only a short sequence of words to work with.

3.2 Predictive Text

Naturally, Predictive Sentence Generation is closely related to the *predictive text* functionality familiar to PDA and cell phone users. It is, perhaps, on these devices where Predictive Sentence Generation might most appropriately be applied. Applications in this area are typically implemented with the intent of mitigating hardware limitations[4][5], though research on further constraining input functionality has been conducted as well[6]. In either case, Predictive Sentence

Generation and predictive text share the common goal of minimizing keystrokes without compromising textual input.

Like Predictive Sentence Generation, the use of a user-produced, growing lexicon is not uncommon in predictive text applications[4][5]. Using this kind of lexicon enables the system to *learn* words it has not encountered before. Of course, this dependence may make such systems untenable on devices where system memory is limited. For the purpose of this experiment, memory constraints brought on by a growing lexicon will not be considered.

One criticism made about simpler predictive text systems is that only the last letter entered is considered when generating word-completion suggestions [5]. In essence, this is the problem addressed here. Rather than simply considering the word immediately to the left of the current, the last *two* words are examined when making sentence-completion suggestions. By expanding the field under consideration, it is hoped that word predictions will be more accurate and therefore more useful to the user.

4 Scope

At its core, this research attempts to improve hardware usability by improving predictive word-suggestion accuracy. Toward this end, one factor of consequence is being scrutinized: the order in which words appear in a sentence. For the purposes of this experiment, only two words are taken into consideration: those immediately to the left of the current cursor position. The goal of this experiment is to determine how setting this constraint affects the accuracy of the Predictive Sentence Generator.

As well, the Predictive Sentence Generator described later only offers five word suggestions, if they exist. This limit was imposed for one reason: the benefits of Predictive Sentence Generation would be of little benefit if it is faster for the user to type a word than it is to pick it out of a lengthy list.

5 Hypothesis

I suspect that narrowing the criteria by which a prediction can be made will improve the accuracy of the Predictive Sentence Generator. Taking two words into consideration instead of just one should lower the number of suggestions returned. It seems natural that this measure would improve system performance. Putting more constraints on possible word-suggestions will likely disqualify some candidates, which may have a high rate of occurrence overall, but are not suitable in the given context.

6 Methodology

6.1 Data

At the highest level, Predictive Sentence Generation attempts to address the same problems addressed by predictive text systems on cell phones and PDAs.

That is, it attempts to provide a software solution to the problem of constrained text-entry hardware. Because of this, it is important to select corpora produced by real people facing constraints similar to those experienced by users of handheld computing devices. Since an examination of data-entry hardware is beyond the scope of this research, the user constraints considered only involve the transfer of data.

Twitter was chosen as the source of experimental data because it satisfies the overall user-oriented focus of this experiment and because it neatly straddles two contexts: handheld and desktop computing. Experiments will be conducted in a desktop environment exclusively, though the limits Twitter places on message length [1] will make the data collected relevant in a handheld environment as well. Of course, it is recognized that for many users message-writing practices vary between desktop systems and their handheld devices. Nonetheless, the Twitter environment is conducive to a highly personalized writing style and is implemented in the relevant contexts.

Individual corporal data generated through Twitter interaction was used to build lexicons useful to the Predictive Sentence Generator. Using an independent ranking system[2], eight individuals were selected based on the number of messages they had submitted to the system. No one with fewer than 3200 entries was selected[3]. The people chosen were: 1) news anchor, Anderson Cooper; 2) Twitter creator, Jack Dorsey; 3) musician/TV personality, MC Hammer; 4) celebrity blogger, Perez Hilton; 5) musician, Questlove; 6) musician, Soulja Boy; 7) TV personality, Tila Tequila; and 8) actor, Wil Wheaton. The corpora assembled from these individuals' Twitter postings were used to both generate lexicons and assess the effectiveness of the Predictive Sentence Generation system.

6.2 Data Preparation

The individual corpora were copied and pasted straight from the Twitter website[4]. Extraneous data (e.g., timestamps, message replies, etc.) was removed with a simple series of Regular Expressions. A tag (*¡tweet¿*) was then inserted between posts in order to facilitated future parsing within the Predictive Sentence Generator. The resulting files were then put into reverse order so that the oldest Twitter posts appeared first. This reversal was necessary in order to maintain the natural flow of the one side of the conversation.

Having removed extraneous data, all of the Twitter users ended up with fewer than the maximum 3200 messages. Often these users would simply relay a message sent to them by a friend. These posts were removed because they were not authored by the party of interest. It is also important to note that total word counts varied wildly. This is because users do not always make use of every one

[1] http://news.cnet.com/8301-17939_109-9677034-2.html?tag=mncol%3btxt

[2] http://twitterholic.com/

[3] It has been determined that Twitter only stores an approximate maximum of 3200 messages, or 160 pages of data

[4] http://www.twitter.com

of the 140 characters available to them. Often messages consisted of only one word.

6.3 Data Structures

There are two notable data objects in the Predictive Sentence Generating system: *Word*s and *Lexicon*s. In short, Words store the relationships between a any given word and its *heads* and *tails* (the words that precede and follow it, respectively). The Lexicon stores the Words and makes *predictions*. Together, Words and Lexicons are the foundation upon which the Predictive Text Generator is built.

The relationships monitored inside a Word object are primarily concerned with *frequency*; how often does a head precede a word and how often does a tail follow it? Words contain one additional hybrid relationship, without which the *headers* would be useless. These are called *neighbours*. Neighbours combine heads and tails in one convenient relationship. Naturally, the frequency with which a neighbour relationship occurs with a word is of especial interest to this experiment. The neighbour relationship is the one that enables the Predictive Sentence Generator to account for word order.

Aside from containing the words that make up the actual lexicon, the Lexicon object embodies three important methods with which to make predictions: 1) Tails Only; 2) Neighbours Only; and 3) Tails & Neighbours. The *Tails Only* method establishes a baseline by which the *Neighbours Only* method is assessed. It only concerns itself with the *last* word in the sentence. Neighbours Only, by contrast, analyzes the last two words. The *Tails & Neighbours* method employs the previous two methods in a further attempt to improve word-prediction accuracy. In this case, both the Neighbours Only and Tails Only methods are polled for word suggestions, but the predictions made by Neighbours Only are given precedence.

6.4 Software & Workflow

The Predictive Software Generator used in this experiment, dubbed *ParsePS*, was designed and written in Java's NetBeans IDE. It allows for hands-on and automated experimentation. There are essentially three modes of operation: 1) Test Mode; 2) Live Mode; and 3) Experimental Mode. *Test Mode* allows the user to assess the viability of the system without making changes to the corpus and lexicon. *Live Mode* offers similar functionality. However, if the *Update* button is pressed in Live Mode, the corpus and lexicon will incorporate any data entered in the test window. *Experimental Mode* is fully automatic. When selected, a file selection window opens, which allows the user to choose a corpus to be processed. Experimental Mode emulates the behaviour of the user who actually built the corpus. Words from the corpus are entered one by one and word suggestions are made and considered. Statistical data measuring the accuracy of the word predictions is collected and saved to file.

New data is processed one message at a time. ParsePS does not incorporate new data into the corpus and lexicon until an *update* is performed (either user-initiated or emulated). Given the signal to update, the incoming string is added to the current corpus file and then tokenized. Each token is then converted to lowercase and stripped of all peripheral non-alphanumeric characters (e.g., *Hello!* becomes *hello*, and *didn't?* becomes *didn't*). Each word is checked against the existing lexicon. If it does not already exist, it is created and head/tail/neighbour relationships are recorded. If it already exists, the aforementioned relationships are updated with new frequency data. The newly processed lexical data is now available to assist in completing the next message.

The lexicon contained within the Lexicon object consists of words that *map* to Word objects. When the user (real or emulated) presses the Space Bar, the software knows to request a list of word predictions from the Lexicon, passing it the two words entered before the *space* (when using anything but the Tails Only method, which only requires the most recent word). These words are then used to retrieve their associated Word object, which contains the frequency relationships necessary to make a prediction. Naturally, relationships that show up with the highest frequency are recommended.

7 Assessment

The impact word order constraints have on Predictive Sentence Generation will be assessed against the baseline established by the same system set free from those constraints. Eight corpora will all be processed with all three methods previously discussed. The output from the Tails Only method sets the baseline for each. The output generated by the Neighbours Only and Tails & Neighbours methods will be assessed against that baseline.

Result assessed against their baselines will provide a sense of *predictive accuracy*. Here, accuracy is the measure of how many times a word suggested by the Lexicon object is accepted by the user or emulator. Thus, if word order constraints have a *positive* impact on Predictive Sentence Generation, then accuracy will exceed the baseline and increase with each new message posted.

The order of the five word suggestions presented by ParsePS have no bearing on the statistics collected. Words are presented in order of likelihood, but no reward or penalty is measured if the suggested word is positioned further down the list.

8 Results

It has been determined that word-order constraints alone have a negative impact on word suggestion accuracy. See Table 1 for a summary of the baseline results. The entries in Table 1 and all others are ordered from least accurate to most accurate in terms of final accuracy. Notice that there is no discernable correlation between the number of words or messages produced by an author and the overall accuracy of the word suggestions.

The results of the Neighbours Only method are summarized in Table 2. Neighbours Only consistently under performed against the baseline. In retrospect, these results should have come as no surprise. After all, an incipient corpus would have no word-order relationships to speak of. By eliminating the Tails Only suggestions as options, ParsePS simply did not have anything to offer the user. As such, word suggestion accuracy suffered.

By combining the baseline Tails Only method with the restrictive Neighbours Only method, a small improvement in overall accuracy was demonstrated (see Table 3). This solved the problems associated with having a very small initial corpus. Since there are very few neighbour relationships in the early stages of development, tail relationships were allowed to makes suggestions where there would have been none otherwise.

Local Accuracy describes the success of a prediction on a given message (e.g., 3 successful word suggestions in a message 10 words long results in 30% accuracy). See Table 4 for an average summary of these results. As with the previous data, the Tails Only method outperformed Neighbours Only, and the combined method outperformed them both.

Table 1. Baseline (*Tails Only*) Statistics with Final Accuracy Results

Corpus Producer	#Msgs.	#Words	Avg. Msg. Length	Final Accuracy
Questlove	3197	41408	12.91492024	0.1094716
MC Hammer	2893	40019	13.83304528	0.13523576
Perez Hilton	3173	32727	10.31421368	0.1415345
Jack Dorsey	3194	29128	9.119599249	0.1435732
Anderson Cooper	3197	23952	7.379418205	0.16514072
Wil Wheaton	3097	58137	18.77203746	0.1674149
Tila Tequila	3197	67137	20.72880826	0.17339171
Soulja Boy	3178	34013	10.70264317	0.18419428

Table 2. *Neighbours Only* Final Accuracy Results

Corpus Producer	Avg. Baseline Distance	Final Accuracy
Questlove	-0.068324885	0.02526082
Jack Dorsey	-0.086791075	0.042879704
MC Hammer	-0.079031857	0.043479346
Perez Hilton	-0.080959707	0.046903167
Anderson Cooper	-0.098238576	0.048533402
Wil Wheaton	-0.087182174	0.0665841
Tila Tequila	-0.095371217	0.073997945
Soulja Boy	-0.081980865	0.08640814

The general success of ParsePS's ability to *learn* new words is illustrated in Figure 1. That graph is typical of all the data produced. Over all, there is a steady improvement in accuracy. That is not to say that accuracy *never* declines from one message to the next, but rather that accuracy improves more often than it declines.

Fig. 1. Software Learning Progress - Soulja Boy

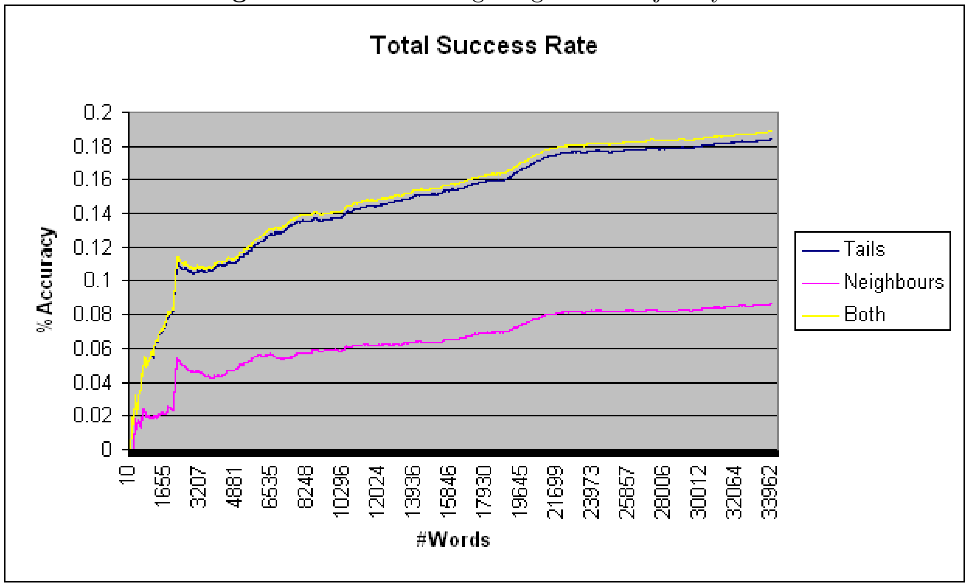

Table 3. *Tails & Neighbours* Final Accuracy Results

Corpus Producer	Avg. Baseline Distance	Final Accuracy
Questlove	0.000987907	0.1100029
MC Hammer	0.001581089	0.13803443
Perez Hilton	0.000975798	0.14269564
Jack Dorsey	0.00060358	0.14515243
Anderson Cooper	0.000140358	0.16594608
Wil Wheaton	0.001754457	0.16922098
Tila Tequila	0.002653663	0.17829216
Soulja Boy	0.003563386	0.18901597

Table 4. Summary of *Local Accuracy* Results

Corpus Producer	Neighbours Only	Baseline (Tails Only)	Tails & Neighbours
Questlove	0.021179437	0.089583325	0.090605895
MC Hammer	0.028843074	0.10787493	0.109456019
Jack Dorsey	0.027192221	0.113983296	0.114586877
Perez Hilton	0.036998987	0.117958694	0.118934492
Anderson Cooper	0.036616755	0.134855331	0.134995689
Wil Wheaton	0.051708498	0.138890671	0.140645128
Soulja Boy	0.065202992	0.147183857	0.150747243
Tila Tequila	0.052127229	0.147591846	0.150250013

9 Conclusion

This experiment was a success. The impact word-order constraints have on Predictive Sentence Generation has been determined. Word-order constraints do improve predictive accuracy when coupled with an unconstrained prediction mechanism. This coupling was implemented in the Tails & Neighbours Method. The constrained mechanism (Neighbours Only), executed alone, under performs when measured against the baseline set by the unconstrained mechanism (Tails Only).

This experiment was met with some constraints of its own. The Twitter corpora were not nearly large enough. It is suspected that given larger corpora, the constrained Neighbours Only method would perform better. This suspicion is fueled by the fact that in the early stages of corpus formation, only considering words with a defined context severely limits the number of suggestions produced. With such a small set of data to work with, the ParsePS system never really surpasses the *early stages* of corpus formation. Nonetheless, the small improvements in accuracy observed in this experiment are encouraging, because accuracy may continue to improve as the corporal body grows.

Finally, predictive accuracy could be further improved upon if predictive text was incorporated into the Predictive Sentence Generator. If letters, or parts of words, were taken into consideration, the list of word suggestions could be constantly reevaluated and refined.

References

1. He, Q., Jiang, D., Liao, Z., Hoi, S.C.H., Chang, K., Lim, E.P., Li, H.: Web query recommendation via sequential query prediction. Data Engineering, International Conference on **0** (2009) 1443–1454
2. Lei, M., Fan, L.: A web personalization system based on users' interested domains. In Wang, Y., Zhang, D., Latombe, J.C., Kinsner, W., eds.: IEEE ICCI, IEEE (2008) 153–159

3. Marcialis, I., Vita, E.D.: Searchy: An agent to personalize search results. Internet and Web Applications and Services, International Conference on **0** (2008) 512–517
4. Hiroyuki Komatus, S.T., Masui, T.: Corpus-based predictive text input. Active Media Technology, 2005. (AMT 2005). Proceedings of the 2005 International Conference on (2005) 75–80
5. van den Bosch, A., Bogers, T.: Efficient context-sensitive word completion for mobile devices. In: MobileHCI '08: Proceedings of the 10th international conference on Human computer interaction with mobile devices and services, New York, NY, USA, ACM (2008) 465–470
6. Dunlop, M.D., Montgomery Masters, M.: Investigating five key predictive text entry with combined distance and keystroke modelling. Personal Ubiquitous Comput. **12**(8) (2008) 589–598