Proceedings of the Second IASTED International Conference
HUMAN-COMPUTER INTERACTION
March 14–16, 2007, Chamonix, France
ISBN Hardcopy: 978-0-88986-654-6 / CD: 978-0-88986-655-3

A WORD PREDICTOR FOR INFLECTED LANGUAGES: SYSTEM DESIGN AND USER-CENTRIC INTERFACE

Carlo Aliprandi
Synthema Srl
Pisa, Italy
email: aliprandi@synthema.it

Nicola Carmignani, Paolo Mancarella and Michele Rubino
Department of Computer Science
University of Pisa, Italy
email: {nicola, paolo, rubino}@di.unipi.it

ABSTRACT
We present FastType, a word prediction system for the Italian inflected language, and its user-centric interface. FastType has greatly evolved from its original features. We have added new linguistic resources, implemented more effecient prediction algorithms and made a brand-new user interface. Thanks to the prediction engine upgrades, like the generation of word and Part-of-Speech *n*-gram collections, and to the introduction of a linear combination algorithm, performances are greatly improved. Keystroke Saving reached 48% and is now comparable to the one achieved with state-of-the-art methods for non-inflected languages. DonKey, the new human-computer interface, allows the user to benefit from automatic word completion in any application. FastType is primarily designed for users with special needs and to reduce misspellings for users with linguistic difficulties.

KEY WORDS
Interaction Design for People with Disabilities, Speech and Natural Language Interfaces, User Interface Development, Word Prediction, Alternative and Augmentative Communication.

1. Introduction

The FastType word prediction system for the Italian (inflected) language has been introduced in [1]. At each keystroke, FastType suggests a list of meaningful predictions, amongst which the user can identify the word he meant to type. By selecting a word from the list, the system will automatically complete the word being written, thus saving keystrokes and helping people with motor impairments as an Alternative and Augmentative Communication (AAC) [2] device. In order to improve the original, naive, interface of FastType, we conducted a survey among impaired students from USID (Unit for Support and Integration services for Disabled students of the University of Pisa) to identify their needs, their wishes for an assistive writing environment and their will to use a word prediction software to speed up text editing.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of the state-of-the-art in word prediction. The improvements of FastType in terms of language resources and algorithms are presented in Section 3. These improvements have led to performance boosting of FastType as shown in Section 4. Section 5 discusses the results of the mentioned survey. DonKey, the new user-centric interface, is described in Section 6. Finally, Section 7 draws some conclusions and describes future work.

2. State of the Art

The main goal of word prediction is guessing and completing what word a user is willing to type in order to facilitate and speed up the text production process. Word predictors are intended to support writing and are commonly used in combination with assistive devices such as keyboards, virtual keyboards, touchpads and pointing devices. Prediction methods have become quite popular in mobile phone softwares. Commercial systems as Tegic Communications T9, Zi Corporation eZiText and Motorola Lexicus iTap are all successful systems that adopt a very simple method of prediction based on dictionary disambiguation. They are commonly referred to as letter predictors.

Word predictors typically use more refined stochastic language models with context information in order to predict a full word instead of a single letter.

Word prediction for non-inflected languages achieved good results. For instance [3] and [4] present language processing techniques that allow the user to save more than 50% of keystrokes thanks to prediction software. The main contribution of FastType is the adaptation and improvement of these techniques for inflected languages. Indeed the latter pose a harder challenge to prediction algorithms than non-inflected languages, due to the high number of inflected forms that dramatically decrease Keystroke Saving (KS). Being Russian and Basque two languages rich in inflectional word forms, [5] and [6] use a morphological component in a two step procedure to compose inflections from the root forms of words. FastType follows instead a one-step procedure, presenting to the user a list of word forms. These forms are correctly inflected by analyzing the sentence context, made up by Part-of-Speech (POS) and related morpho-syntactic information about the previous words. This procedure, combined with on-the-fly POS tagging, enables FastType to perform a KS comparable to the one achieved for non-inflected languages.

3. Upgrading the Prediction Engine: New Language Resources and Algorithms

The architecture of FastType can be described as follows:

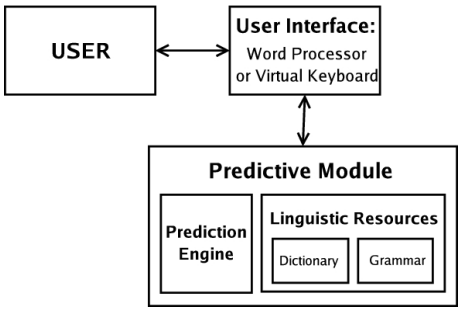

Figure 1. FastType Architecture

As shown in Figure 1, FastType has three main components: the *User Interface*, the *Predictive Module* and the *Linguistic Resources*. The *Prediction Engine* is the kernel of the *Predictive Module*; it manages the communication to and from the User Interface, keeping trace of the prediction status and of typed words. At each keystroke it predicts suggestions, in the form of list of word completions and of list of next-letters, by assuring accordance (gender, number, person, tense and mood) with the syntactic sentence context. The *Predictive Module* functionalities, such as the morpho-syntactic agreement and the lexicon coverage, are provided by the statistic language model based on POS tag *n*-grams and on morphological information provided by the *Linguistic Resources*.

3.1 New Language Resources

A basic assumption in word prediction is that contextual information affects the environment where the word has to be entered. In order to predict the most likely word it is necessary to have a high-order representation of the context. This assumption has been consolidated into stochastic methods that are based on Markov models and n-gram word models. The task of word prediction can be modeled as the estimation of probability to guess the N$^t h$ word (w_N) given the current sequence of N-1 previous words, denoted by

$$\mathbb{P}(w_N \mid w_1, w_2, \ \ldots \ , w_{N-1})$$

Usually, as the vocabulary size is very large, it is necessary to approximate *n*-gram models by unigrams ($n = 1$), bigrams ($n = 2$) and trigrams ($n = 3$):

$$\mathbb{P}(w_i \mid w_1, \ldots, w_{i-1}) \approx \mathbb{P}(w_i \mid w_{i-n+1}, \ldots \ , w_{i-1})$$

N-grams of POS are used as a function $\wp$ to restrict the increase of the context parameters into an equivalence class, so that

$$\mathbb{P}(w_i \mid w_1, \ldots, w_{i-1}) \approx \mathbb{P}(w_i \mid \wp[w_{i-n+1}, \ldots \ , w_{i-1}])$$

We improved the original POS trigram model by means of a *trigram unification algorithm*. In the original model every word is tagged with a POS tag containing deep morpho-syntactic information. As an example, the Italian sentence "io ti amo" ("I love you") is tagged as io[NPN1S1(io)] ti[NPN2S2(tu)] amo[VTN1IN(amare)], that is:

- NPN1S1, a first-person personal pronoun;
- NPN2S2, a second-person personal pronoun;
- VTN1IN, a present indicative first-person verb.

The first kind of unification is *generalization* and can be shown in the following example. During its training, FastType learns POS trigram sets which are very specific, e.g. "voglio scrivere lettere" ("I want to write letters") is modeled into the POS trigram "VNN1IN VTNIFN SCFPFS", while "voglio scrivere libri" ("I want to write books") is modeled into "VNN1IN VTNIFN SCMPMS". The above two trigrams differ only in the last unigram for the gender tag (F vs M). Since in this case the gender info is not relevant for syntactic correctness, the generalization merges the two trigrams into a single `new' trigram "VNN1IN VTNIFN SC0P0S". In the last unigram the 0 is considered as a wildcard telling the engine to ignore gender information when selecting suggestions.

A second kind of unification is *parameterization*. During training, FastType learns POS trigrams where gender or number information are always in accordance. For example "il mio gatto" and "la mia gatta" (in Italian "gatto" is a male cat and "gatta" is a female cat) generate the POS trigrams "RDMSMSGEMSMSSCMSMS" and "RDFSMS GEFSMS SCFSMS". Since there is a gender agreement, the unification algorithm can merge the trigrams into a single one, "RD∗SMS GE∗SMS SC∗SMS". As opposed to the 0 wildcard, here ∗ stands for a logical variable, that is reference-resolved at run time: if the user writes "la mia", the two words are classified as feminine singular and the ∗ for gender is resolved as F, and thus only feminine singular nouns (like "gatta" and not "gatto") will be suggested.

Unification brought FastType two considerable improvements, a more generic word search and an additional 8% of KS. Moreover, the lower number of POS trigrams (the original set of 76.000 trigrams dropped to 19.000) greatly increased the speed in searching suggestions.

To further improve the prediction engine, two new language resources were created: POS bigrams and Tagged Word (TW) bigrams. POS bigrams and TW bigrams were trained from a large balanced corpus (approximately 2.000.000 word forms) created from newspapers, magazines, documents, commercial letters and e-mails. The corpus was cleaned, standardized (punctuations, capitalizations) and then parsed using a rule-based parser, the Italian POS tagger Synthema Lexical Parser (SLP) [7]. The re-

sult was a corpus tagged with syntactic and morphological knowledge that was used for training.

POS bigrams can capture context information for pairs of words. They are particularly useful for word prediction at the beginning of sentence and in short sentences.

POS bigrams are also used as a backup model for POS trigrams: the engine, rather than accepting a POS trigram with a probability under a given threshold, extends the search to POS bigrams, in order to get a more significant POS. A higher probability produces better suggestion ranking (see Section 3.2).

Finally, we introduced a new language resource, TW bigrams: the typical Bigram model estimates the probability of a word given the preceding one:

$$\mathbb{P}(w_i \mid w_{i-1})$$

We extended this model by adding POS information. A word bigram (w_{i-1}, w_i) is extended to (w_{i-1}, w_i, t_i), where t_i is the POS of w_i. The probability of (w_{i-1}, w_i, t_{i-1}) is estimated by

$$\mathbb{P}(w_i \mid w_{i-1},\ t_i)$$

In the next section we show how this new resource is fundamental for the Linear Combination ranking algorithm.

3.2 New Prediction Algorithm

We introduce a new prediction algorithm based on Linear Combination extending the baseline algorithm presented in [1]. The approach closest to ours is the one presented in [3], that is a Linear Combination algorithm combining POS trigrams and simple word bigrams.

Our Linear Combination extends this model to cope with inflected languages, by combining the POS trigram model with the two new language models previously described in this section. The POS trigram model finds the most likely POS tags for the current word, given the two previous POS tags, if necessary backed up by POS bigrams. The TW bigram model finds the most likely words given the immediately preceding word. The probability S for the current word is the result of a weighed combination of the models:

$$S = x \cdot \mathbb{P}(w_i \mid w_{i-1}, t_i) + y \cdot f(t_i, t_{i-2}, t_{i-1})$$

where

$$f(t, t', t'') = \begin{cases} \mathbb{P}(t \mid t',\ t'') & \text{if } \mathbb{P}(t \mid t',\ t'') > \vartheta \\ \mathbb{P}(t \mid t'') & \text{otherwise} \end{cases}$$

and ϑ is the threshold. x and y are the coeffecients of the linear combination and their sum must be 1 ($x + y = 1$).

To measure FastType performance improvements with the new Linear Combination algorithm we ran trials on the same test set presented in [1], which is composed of 40 texts randomly selected from Web news and journal papers. Notice that the test set is completely disjoint from the training set.

We developed a new test bench, performing different trials to experimentally determine the optimal value for x and y. The nutshell of the test bench is a `simulated user' typing the test set and acting as a user that always selects the correct suggestion when predicted. We then measured the KS varying values for x and y. We ran trials increasing x by $0,1$ from $0,1$ to $0,9$. The best KS was obtained for $x = 0,6$.

The new Linear Combination algorithm, raised considerably KS. Our optimal test bench produced an average KS of 45% with a prediction list of 10 words. The KS for the baseline algorithm, presented in [1] was 30%. We believe this is an outstanding result for an inflected language as we detail in Section 4.

4. FastType Performance Measurements

FastType has been tested to appraise its utility. We have choosen to evaluate the system with the respect to the following three performance measures. The first two are commonly found in literature while the third one has been introduced to compare the time saving with the "typing effort" saving expressed by KS.

1. ***Keystroke Saving*** (KS): the percentage of keystrokes the user "saved" by using FastType. Being C_t the number of keystrokes needed to write a text without FastType and C_d the number of keystrokes needed to write the same text with FastType

$$KS\ =\ \frac{(C_t - C_d)}{C_t}\ \times\ 100.$$

2. ***Keystrokes until Completion*** (KUC): being $c_1\ \ldots\ c_n$ the number of keystrokes for each of the n words before the desired suggestion appears in the prediction list

$$KUC = \frac{(c_1 + c_2 + \ \ldots\ + c_n)}{n}.$$

3. ***Word Type Saving*** (WTS): the percentage of time the user saves with FastType. Being T_n the time needed to write a text without FastType and T_a the time needed to write the same text with FastType

$$WTS = \frac{(T_n - T_a)}{T_n} \times 100.$$

We developed a simulated writer that typed the test set discussed in Section 3. The presence of a simulated user is needed to guarantee constant typing speed, which is fundamental to calculate WTS percentages. A real life test with human writers is planned for future work, as shown in Section 7. A parameter that can greatly influence performance measurements is the length L of the prediction list, so we

L	KS	KUC	WTS
5	41,15%	2,85%	21,12%
10	45,26%	2,67%	24%
20	47,9%	2,48%	24,35%

Table 1. Performance Measurement Results

ran three trials on the test set with $L = 5$, $L = 10$ and $L = 20$. Results are shown in the Table 1.

As we can see in Table 1, the increase in KS, WTS and KUC between $L = 5$ and $L = 10$ is way more relevant than the increase between $L = 10$ and $L = 20$. This result was relevant to design the graphical user interface we discuss in Section 6. Performances are significantly better than existing works on inflected languages, that achieve a KS of 30%. With $L = 10$, KS for the improved FastType rises to 45%. A graphical example of keystroke saving with L = 10 is shown in Figure 2 where saved characters are marked in gray.

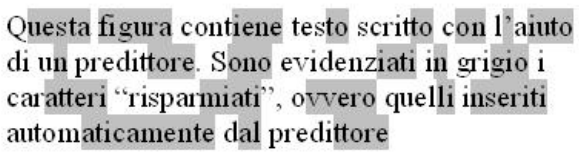

Figure 2. Sample text written with the help of FastType

5. Survey Results

A survey among impaired students from USID was conducted to guide the design of the new human-computer interface. The results outlined that:

- Most users, as their primary computer activity, write text for study and work (two very time-consuming activities which occupy the largest part of the day).
- Most users communicate intensively through the Internet by means of written text (mainly e-mail). This, in conjunction with the previous item, is evidence of the importance of writing text in their everyday life.
- Despite their handicap, almost all users prefer the standard hardware keyboard to write text and work with the computer. Even eyesight-impaired users, who like the Braille bar to get feedback from the computer, choose the standard keyboard to write. Some users prefer a virtual keyboard or another assistive input software that allows text input via the mouse. Automatic Speech Recognition is uncommon but known to some users. A word prediction interface should allow choosing suggestions via both mouse and keyboard and should also include speech recognition to choose the desired suggestion.
- Microsoft Office® is most commonly used for text editing, so compatibility with it is needed for a word prediction interface.
- Most users would appreciate an assistive writing software.
- Text-to-Speech and audio feedback would be greatly appreciated in a word prediction interface.
- The users state that, regardless of the underlying technology for word prediction, the system interface should be kept as simple as possible.
- The typing speed of most users is up to 100 characters per minute (though some of them can reach 200 characters per minute). This giving us a time upper bound of about 0,5 seconds to fill the prediction list. A slower prediction engine would in the end slow down the user's writing, which is not desirable.
- About half of the users need a specific dictionary for their work (or study); in such contexts a prediction software must be designed to support domain-oriented dictionaries.

6. A New Interface: DonKey

The analysis of the survey we conducted was used to design and develop a new user interface (called *DonKey*) that nicely complements the prediction engine. Since nearly all users use keyboard and mouse to input text into MS Office programs, we built an interface that smoothly integrates into MS Windows where users can choose suggestions for automatic completion with mouse and keyboard. Furthermore, we made sure it would be easy to understand and use.

The user interface we created is shown in Figure 3. Donkey is predicting words for the current uncompleted word ("*m...*") after a feminine singular article ("*la*") and a feminine singular adjective ("*mia*"). Notice that only feminine singular nouns are predicted ("*madre*", "*mamma*", "*moglie*", etc.) whereas masculine or plural nouns are dropping out the prediction list. This morpho-syntactic accordance is of the utmost importance for inflected languages. Plus, by implementing a "sieve" function on nouns, it greatly simplifies the suggestion ranking to ensure that the most likely words will appear in the list.

DonKey works in conjunction with every MS Windows application, interacting with peripherals and external devices. While the user writes text into e.g. MS Word, DonKey catches the keystrokes and feeds them to the prediction engine to build up its internal status and sentence context. The engine returns suggestions for word completion to the interface and the user can select them either by

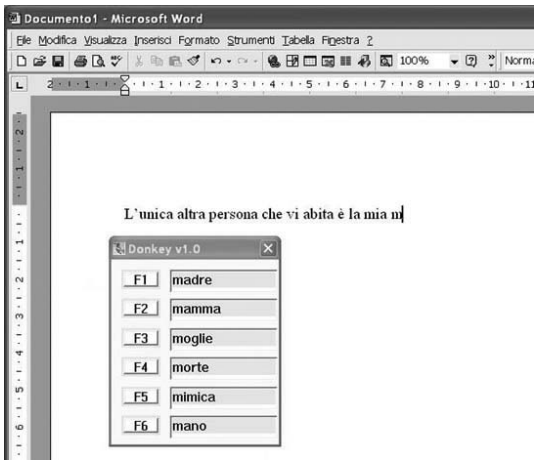

Figure 3. The DonKey Interface

pressing the corresponding function key or by clicking on the corresponding button. Figure 3 shows a screenshot of DonKey where six words are suggested to the user. However he can customize the number of suggestions by choosing the desired value in the range of 1-10 (selectable with F1 to F10 function keys). When the user hits a function key (or clicks on the corresponding button on the interface), DonKey sends back the corresponding suggestion, thus completing the current word. We are currently working on including a virtual keyboard with integrated prediction and automatic speech recognition, to fulfil users' needs.

DonKey also features audio feedbacks to help eyesight-impaired writers. Sounds warn the user, e.g. when no suggestions have been found as well as to confirm that the word selected has been inserted. Another kind of audio feedback is Text-to-Speech (TTS). DonKey plays natural speech feedback to the user. The TTS module can be configuredto speak aloud given suggestions and the selection made by the user. Speech feedback is clearly alternative to audio feedbacks. The user can easily configure DonKey by activating or deactivating speech feedback, by changing the voice parameters (like speed) and setting the desired number of suggestions. More suggestions provide higher KS, less suggestions provide easier and faster reading. Our test bench showed that a suggestion list of length 10 (default value) gives already good performances, while a longer list would bear an excessive cognitive load on the user.

The configuration program should be as easy to understand and use as DonKey, so we avoided complex and cumbersome configurations, like lots of options, a tiny character set (hard to read) and technical terms and choices that can be difficult to make for a computer-layman. We developed a very user-friendly configuration program, with few, easily understandable options in a single dialog window. TTS confirms users' settings.

In short, FastType provides a simple interface, which is particularly easy to use.

7. Conclusion and Future Work

We presented various improvements of the FastType prediction engine, which made it a real prediction software for inflected languages. Indeed, according to our tests, the system performances are comparable to those of predictors for non-inflected languages. The major improvements have to do with new linguistic models based on new linguistic resources, and with an extended Linear Combination algorithm.

The system reaches a KS up to 45%, which is comparable to the KS achieved by others for non-inflected languages. Performances are significantly better than existing works on inflected languages, that achieve a KS of 30%. The system is also equipped with a versatile, easy to use and user-centric interface that will be used in the planned user evaluation. In the next year FastType will be betatested by disabled users from USID to measure their satisfaction in real usage.

Further work is planned to establish additional improvemens in terms of flexibility, accessibility and prediction quality. First of all we intend to investigate the system adaptability to the user's style in order to enrich FastType with a personal language model and a personal dictionary. FastType will feature machine learning of "writing styles", and the prediction engine will learn not only the user's lexicon but also the sentence context (i.e. the user writing style).

Combining existing methods with the user language model and dictionary, we expect to obtain, after a starting training period, a more effective prediction user by user. FastType will include ASR to provide the user with a more simple and free interaction. We will assess real bene□tsof ASR applied to word prediction. Moreover we are planning to design and develop a word prediction interface for portable devices (handheld PC, PDA, smartphones) and, if necessary, an appropriate "portable device version" of the prediction engine.

Acknowledgement

The FastType project is partially funded by the Fondazione Cassa di Risparmio di Pisa.

References

[1] C. Aliprandi, N. Carmignani and P. Mancarella, An Inflected-Sensitive Letter and Word Prediction System, *Proceedings of the International Conference on Interactive Computer Aided Learning,* Villach, Austria, 2006.

[2] A. Copestake, Augmented and Alternative NLP Techniques for Augmentative and Alternative Communication, *Proceedings of the ACL Workshop on Natural Language Processing for Communication Aids*, Madrid, Spain, 1997, 37–42.

[3] A. Fazly and G. Hirst, Testing the Efficacy of Part-Of-Speech Information in Word Prediction, *Proceedings of the 10th Conference of the European chapter of the Association for Computational Linguistics*, Budapest, Hungary, 2003.

[4] T. Miller and E. Wolf, Word Completion with Latent Semantic Analysis, *Proceedings of the 18th International Conference on Pattern Recognition (ICPR '06)*, Hong Kong, China, 2006, 1252–1255.

[5] S. Hunnicutt, L. Nozadze and G. Chikoidze, Russian Word Prediction with Morphological Support, *5th International Symposium on Language, Logic and Computation*, Tbilisi, Georgia, 2003, 91–96.

[6] N. Garay-Vitoria and J. Abascal, Word Prediction for Inflected Languages. Application to Basque Language, *Proceedings of the ACL Workshop on Natural Language Processing for Communication Aids*, Madrid, Spain, 1997, 29–36.

[7] R. Raffaelli, Lexical Data Base Management System – LDBMS, *Synthema Internal Report*, Pisa, 2000.