IJCLA **ISSN 0976-0962**

International Journal of Computational Linguistics and Applications

Vol. 3 **No. 2** **Jul-Dec 2012**

Guest Editor
Yasunari Harada

Editor-in-Chief
Alexander Gelbukh

ISSN 0976-0962

International Journal of Computational Linguistics and Applications

Vol. 3 **No. 2** **Jul-Dec 2012**

International Journal of Computational Linguistics and Applications – IJCLA (started in 2010) is a peer-reviewed international journal published twice a year, in June and December. It publishes original research papers related to computational linguistics, natural language processing, human language technologies and their applications.

The views expressed herein are those of the authors. The journal reserves the right to edit the material.

Indexing: Cabell's Directory of Publishing Opportunities.

Editor-in-Chief: Alexander Gelbukh

Subscription: India: Rs. 2699
Rest of the world: US$ 249

Payments can be made by Cheques/Bank Drafts/International Money Orders drawn in the name of BAHRI PUBLICATIONS, NEW DELHI and sent to:

BAHRI PUBLICATIONS
1749A/5, 1st Floor, Gobindpuri Extension,
P. O. Box 4453, Kalkaji, New Delhi 110019
Telephones: 011-65810766, (0) 9811204673, (0) 9212794543
E-mail: bahrius@vsnl.com; bahripublications@yahoo.com
Website: http://www.bahripublications.com

Printed & Published by Deepinder Singh Bahri, for and on behalf of **BAHRI PUBLICATIONS**, New Delhi.

International Journal of Computational Linguistics and Applications

Vol. 3 **No. 3** **Jul-Dec 2012**

CONTENTS

Editorial

This issue of the International Journal of Computational Linguistics and Applications is divided into four topics: quantitative linguistics, lexical resources, parsing and co-reference, and applications. Since one of the papers is twice longer than an average paper in this journal, this issue contains only nine papers and not ten as usually.

The first section of this issue, which includes only one paper, is titled "Quantitative linguistics". Quantitative linguistics includes, among other directions, the study of statistical distributions of letters, morphs, words, sentences and their characteristics. Perhaps the most widely known law of quantitative linguistics is Zipf's law, which relates the frequencies of words in a text with their frequency rank.

M. Perakh (USA) presents an application of serial correlation statistics to the study of meaningful texts. He shows that certain regularities of the distribution of letters are present only in meaningful texts and are not present in meaningless strings of characters. Those regularities are observed in different languages of different language families and with different writing systems. Perakh also reveals the relation between serial correlation statistics and the Zipf's law. I believe that his research can open new perspectives in a number of long-standing research questions, from the study of the Voynich manuscript to deciphering ancient scripts to, maybe, the search for messages of extraterrestrial intelligence. Unfortunately, he did not have a chance to develop and apply this research: this prominent scientist, talented writer and outstanding fighter for democracy passed away before he could finish this paper, which is presented to the reader in the version copyedited by the Editor-in-Chief.

The next section is devoted to lexical resources. Lexical resources are crucial for development and for evaluation of computational linguistics research, providing the empirical basis for the theories and techniques created in frame of all other research directions.

V. Henrich et al. (Germany) present a method for collecting sense-annotated corpora from open Internet. Sense-annotated corpora are very

important for, for example, training supervised word sense disambiguation classifiers, given that supervised techniques proved so far to be more accurate than unsupervised ones, and easier to implement and maintain. The authors show that their method is language-independent: they successfully apply their method to English and German. The two obtained corpora (English corpus annotated with the WordNet sense inventory and German corpus annotated with the GermaNet sense inventory) are freely available to download.

S. Wang (China) continues the topic of WordNet with a discussion of the perspectives of its translation and use in languages other than English, in this case Chinese. During last two decades the WordNet dictionary proved to be very successful in numerous applications, both research and practical; many existing tools and techniques rely on the WordNet structure and sense inventory. However, despite numerous attempts and long-term efforts, the problem of its translation into other languages has not been solved. Similar dictionaries do exist for a number of major languages, but they are not interoperable with WordNet-based tools; as we have seen, the authors of the previous paper used GermaNet for processing German data—even if German is the language most closely related to English. Wang describes the process of translation of English WordNet into a very different language, Chinese.

K. Kotani et al. (Japan) report the creation of first text corpus that contains material reflecting all four modalities of learners of English as foreign language: writing and speaking, in the form of essays and speech by non-native speakers of English, as well as reading and listening, in the form of student's answers to questionnaires on the texts that they read in English or stories that they listened. Such a corpus will no doubt prove very useful in identifying patterns in students' performance, errors and difficulties. The authors discuss the methodology for the selection of the material for this corpus.

R. Kumar et al. (India) present a tool for manual computer-aided annotation of words in texts with part-of-speech tags. Manually annotated corpora are the raw material for both supervised learning of rules for automatic annotation and manually detecting regularities and building corresponding theories. The tool presented by the authors permits to annotate manually all words in the text, while automatically presenting to the user the most probable variant of such annotation. The authors study the effect of such automatic hints on the accuracy and

efficiency of manual annotation. The tool works with Hindi, the world's second largest language.

The next section presents papers devoted to parsing and co-reference detection. Parsing is the task of identifying the internal structure of sentences, the relations between words in the sentences. While the best studied parsing technique is constituency parsing (grouping words together, and grouping such groups together), an alternative approach, dependency parsing (subordinating words to each other: some words in the sentence add more details to other words, making their meaning more specific) gains increasing attention of the research community, especially when dealing with free word order languages. Both papers in this issue devoted to parsing consider the dependency approach.

R. Alfared and **D. Béchet** (France) address the problem of efficiency of a parser by restricting the set of the possible part-of-speech marks of the input words using a separate part of speech (POS) tagger. Given that parsing is a slow operation, the usefulness of parsers for large-scale analysis of Internet texts crucially depends on their speed. The authors show that using a POS tagger significantly increases the parser's speed, while slightly decreasing its recall: the parser misses some correct analyses. The experiments were performed on a French categorial dependency parser.

R. Goutam and **B. R. Ambati** (India) explore the effect of two bootstrapping techniques—self-training and co-training—on a dependency parser, using Hindi as case study. Self-training and co-training are simple variants of semi-supervised learning: the use of unlabeled examples to improve supervised learning techniques. The authors use for their experiments two major Hindi parsers, and compare their results with a those achieved at a competition of Hindi parsers. The authors show that in-domain self-training and co-training gives significant improvement in accuracy, while out-of-domain self- and co-training is less advantageous.

Y. Guo et al. (China) address the topic of entity linking, which can be roughly understood as co-reference. They link named entities found in the text to sources of structured knowledge, such as databases. They use rich context available for the named entity in different texts where it is mentioned to build a model of the entity, so that it can be linked to a correspondent database entry. Using two different benchmark datasets, the authors show that their approach outperforms existing state-of-the art approaches.

The last section of this issue, also consisting of one paper, is devoted to applications.

T. A. Pirinen et al. (Finland) address the problem of spell-checking, probably one of the oldest applications of natural language processing and still far from complete solution. They present a context-aware spell-checker, capable of re-ranking correction suggestions generated by a simpler spell-checker, basing on the information provided by a part of speech tagger. They also show how to adapt traditional *n*-gram models for part-of-speech tagging to morphologically rich languages, with the case study of Finnish, which is an agglutinative language with very rich morphology.

I expect that the papers published in this issue would be useful for scholars, students, and general public interested in natural language processing, applied linguistics, and language learning.

GUEST EDITOR:

YASUNARI HARADA
PROFESSOR,
WASEDA UNIVERSITY, JAPAN
DIRECTOR,
INSTITUTE FOR DIGITAL ENHANCEMENT OF COGNITIVE DEVELOPMENT
PRESIDENT,
ENGLISH LANGUAGE EDUCATION SOCIETY OF JAPAN
EX-PRESIDENT,
LOGICO-LINGUISTICS SOCIETY OF JAPAN
E-MAIL: < HARADA@WASEDA.JP >

Quantitative Linguistics

IJCLA VOL. 3, NO. 2, JUL-DEC 2011, PP. 11–43
RECEIVED 16/09/12 ACCEPTED 18/12/12 FINAL 23/12/12

Serial Correlation Statistics of Written Texts

MARK PERAKH[1]

California State University Fullerton, USA

ABSTRACT

Serial correlation statistics has been widely used in various fields of science, but apparently has not yet been applied to the analysis of texts. In this paper a method is offered using measurements and computations of certain statistical sums that reflect the variability of the letters' distribution along texts. It opened a way for the analysis of texts' structure not available by other means and thus led to the discovery of hidden regularities in the structure of semantically meaningful texts, including, for example, an "average domain of minimal letters variability," common for all semantically meaningful texts in various languages, but absent in meaningless strings of symbols. Another revelation was the connection of certain serial correlation parameters with Zipf's law.

KEYWORDS: *quantitative linguistics, Zipf law.*

1 Introduction

Serial correlation statistics (also referred to as autocorrelation) is widely used in such diverse areas as, for example, econometry [1], spectroscopy [2], or even in music recording [3], and in many other areas. However, to the best of the author's knowledge, it has not yet been applied to the analysis of texts. In this paper a method is described

[1] Mark Perakh passed away soon after submitting this paper, his last publication. The text was copy-edited and formatted later; errors inadvertently introduced in this process are responsibility of the editor.

making use of the serial correlation, which in this case will be dubbed *Letter Serial Correlation* (LSC). It turned out to be a rather powerful tool leading to the discovery of hitherto unknown features of the texts's intrinsic structure.

It is reasonable to assume that meaningful texts possess a certain degree of order. The entropy of meaningful texts is expected to be somewhere between the low entropy of highly ordered meaningless strings and the high entropy of chaotic meaningless strings.

Entropy, though, characterizes the overall level of the disorder in a text but does not reveal the specific features of a text's structure. Therefore it is desirable to develop methods for analyzing specific forms of order in texts.

Imagine that we try to decipher a text written in an unknown language. First we have to determine whether the string of symbols in question is a meaningful text or is gibberish. Information theory is not helpful in this case because its tools are indifferent to the semantic contents of the text. The method of strings' analysis developed in the Algorithmic Probability/Complexity theory [4, 5, 6], while adding a powerful tool to the arsenal of mathematics, linguistics, biology and other fields of inquiry, leaves out the problem of distinguishing between meaningful texts and gibberish. Recent developments in this area [7], while introducing certain markers of noise vs. meaningful messages, do not seem suited to deciphering texts in unknown languages.

In this paper a method for unearthing certain specific structural properties of texts is suggested. It has revealed hidden regularities in meaningful texts' structures. These regularities happen to be present in a wide variety of languages that use alphabetical systems of writing. This method uses a statistical approach based on the analysis of the *variability of symbols' distribution along the string.* It will be referred to as the Letter Serial Correlation statistics, or simply LSC.

2 Basics of the LSC Method

Imagine a string N symbols long. The symbols can be, for example, letters drawn from an alphabet that comprises Z different letters. It can be a text in English, say the *Song of Hiawatha* by Longfellow, wherein N = 141,399 and Z = 26; it can be the German text of any of Goethe's novels where Z = 26 and N varies from novel to novel. It can be the

Hebrew text of the *Book of Genesis*, which is N = 78,064 letters long, with Z = 22. It can be a computer program written as a string of zeros and ones, so Z = 2. It can even be a biological macromolecule wherein each "letter" is a specific chemical compound, etc.

There are three versions of the LSC method. However, of the three versions one turned out to be most informative, therefore in this paper only the data obtained by that version are reported.

When we say that the text's length is found to be N letters long, this number excludes spaces between the words and punctuation marks. We divide the text into equal *cells*, each n letters long. If N is divisible by n, then the number k of cells will be $k = N / n$. If, though, N is not divisible by n, then the last cell at the end of the text will be shorter than the rest of the cells. If k is the number of the "full" cells, each of the same size n, then the total number of cells, including the partial cell at the text's end, will be $r = k + 1$. In such cases the last, partial cell will be cast off and not accounted for.

Let us denote the length of the truncated text, that is the length remaining after casting off the partial end cell, expressed in the number of letters, as L. Obviously, if N is divisible by n, $L = N$, and $k = r$, otherwise $L = kn < N$.

Let us count how many times each letter of the alphabet appears in the entire text, and denote these numbers as M_i ,where the index i takes the values between I = 1 (for the first letter of the alphabet) and I = Z (for the alphabet's last letter).

Let us assign to the cells, remaining in the text after truncation (if such was necessary) numbers from j = 1 (starting at the text's beginning) to $j = k$.

Denote by $X_{i,j}$ the number of occurrences of letter x_i in the cell number j and by $X_{i,j+1}$ the number of occurrences of the same letter x_i in the neighboring cell number $j+1$. Consider the expression $(X_{i,j} - X_{i,j+1})^2$. Squaring the difference ensures the independence of the calculated quantity on whether the letter x_i occurs more often in cell j or in cell $j + 1$.

Comment. Obviously, each cell contains a n-gram. Therefore, some readers may get the impression that we deal here with n-gram statistics. In fact, the serial correlation statistics is quite different from a n-gram statistics. A couple of simple examples may help to see this difference. Let us choose n = 3. Then each cell contains a trigram. Consider a pair of neighboring cells, one containg the trigram [abc] and the other the trigram [def]. What if we shuffle the letters in the cells, getting now a

pair of cells containing, one the trigram [acb] and the neighboring cell containing the trigram [efd]? From the viewpoint of the trigram statistics, the trigrams [abc] and [acb], as well as [def] and [efd], are different trigrams and should be treated as such as long as the trigram statistics is applied. On the other hand, within the serial correlation statistics there is no difference between the cells containing either trigram [abc] or trigram [acb]. Indeed, the expression $(X_{i,j} - X_{i,j+1})^2$, which is at the core of the letter correlation statistics, does not depend on the order of letters within the cells. Letter correlation statistics is concerned with the *variability of letters along the string* and is indifferent to the fact that cells contain n-grams.

Another example of the difference between the approaches of the n-gram and the serial correlation statistics is as follows: the n-gram statistic is only interested in such n-grams which can happen in the explored texts. For example, the trigram [zth] normally does not happen in English texts and therefore it is of no interest for n-gram statistics. Imagine, though, the following string is found in some text: "*The word 'heart' in German is 'Her**z'. Th**is translation can be found in a dictionary.*" Choose n = 3. Then it can happen that one of the cells will contain the following combination of symbols: [z'.(space)Th]. From the viewpoint of the serial correlation, where spaces and punctuation marks are ignored, this combination is equivalent to a cell containing the trigram [zth], and is a legitimate element of the serial correlation statistics.

Now define the following sum, which is referred to as the *Measured Letter Serial Correlation* (LSC) sum:

$$S_m = \sum_{i=1}^{Z} \sum_{j=1}^{k-1} \left(X_{i,j} - X_{i,j+1} \right). \qquad (1)$$

The first summation in equation (1) is performed over all letters of the available alphabet, from $I = 1$ to $I = Z$. The second summation is over all *pairs of neighboring cells*, numbered from $j = 1$ to $j = k - 1$. (Each cell, except for cells number 1 and number k, appears twice in the equation, once paired with the preceding cell and once paired with the subsequent cell; the number of boundaries between the cells, which also is the number of *pairs* of neighboring cells, is $k - 1$).

If measured on a specific text and calculated by equation (1), the sum S_m statistically estimates the variability of letter distribution along the text, averaged over its length.

The interpretation of the behavior of S_m can be facilitated if it is compared with the *Expected Letter Serial Correlation sum*, to be denoted S_e. For a randomized text S_e can be calculated exactly. When calculating the expected letter serial correlation sum, a perfectly random text must be distinguished from the texts obtained by permutations of letters of a meaningful text. In a perfectly random text each letter of the available alphabet has the same probability of appearing at any location in the text. On the other hand, in a text obtained by a permutation of a meaningful text, the frequency distribution of letters is the same as in the original text (the latter to be also referred to as the identity permutation). Therefore in the permuted texts the probabilities of appearing at a certain location in the text are different for each letter.

For example, in English, German, and Spanish texts the most frequent letter is *e* (which in sufficiently long English texts usually occupies about 12 percent of the text). Hence, in a gibberish text obtained by permutation of, say, a sufficiently long English text, the letter *e* will also appear at approximately 12 percent of the locations, so the probability of that letter appearing at an arbitrary location is about 0.12. For the least frequent letter, z, the probability in question is only a fraction of one percent. On the other hand, in a perfectly random text, using the same 26 letter-long alphabet, the probability in question for both *e* and *z* is the same, about 1 / 26.

If a certain letter appears M times in the identity permutation, it will also appear M times in any permuted version of the text in question. On the other hand, this letter, as well as any other letter of the alphabet in use, will appear close to N / Z times in a perfectly random text of the same length of N letters.

In view of the above, the calculation of the expected letter serial correlation sum must be conducted differently for the texts obtained by permutations of a meaningful text and for perfectly random texts. However, the pertinent calculation has revealed that the formulae for S_e, derived for texts randomized by permutation and for a perfectly random text, differ only by the factor $L / L - 1$, where L is the total number of letters in the text (truncated when necessary as described above). Since the studied texts comprised at least several thousand letters each, the above factor was practically equal to 1, so the quantitative difference between expected LSC sums calculated for texts randomized by permutations of letters of a meaningful texts and the sums for perfectly random texts turned out to be negligible.

The expected letter serial correlation sum is calculated by the following equation (derived in Appendix 1):

$$S_e = 2\left(1-\frac{n}{L}\right)\sum_{i=1}^{Z} M_i \frac{L-M_i}{L-1}. \quad (2)$$

The summation in equation (2) is performed over all letters of the alphabet in use.

For the texts subjected to the study, both the measured letter serial correlation sum (as per equation 1) and the expected letter serial correlation sum (calculated by equation 2) are determined for a series of values of the cell size n. This results in two sets of data, one representing the functional dependence of S_m on n, and the other of S_e on n.

These data carry information about the text's structure insofar as it is reflected in the variability of letters distribution along the text.

In many cases it turns out useful to study letter serial correlation utilizing, besides LSC sums, also certain auxiliary quantities. One such quantity is what will be called *Letter Serial Correlation density*. This quantity is obtained by dividing the LSC sums by the cell size n. We distinguish between the measured LSC density d_m, and expected LSC density d_e. For example, the expected LSC density is calculated as

$$d_e = 2\left(\frac{1}{n}-\frac{1}{L}\right)\sum_{i=1}^{Z} M_i \frac{L-M_i}{L-1}. \quad (3)$$

Since LSC densities are obtained from the data on LSC sums, they can't provide information beyond that inherent in the LSC sums. However, in certain cases reviewing the data for LSC densities makes it easier to interpret the observed data. Furthermore, the use of LSC densities revealed the connection between the LSC and Zipf's law [8], as will be shown later in this paper.

Another auxiliary quantity is what will be called *specific letter serial correlation sums*. This quantity is obtained through dividing the LSC sum (either the measured or the expected) by the truncated text's length L. Since in the specific LSC sums, unlike the original LSC sums, the possible effects of the difference in the text's lengths are eliminated, the specific sums are useful if texts of various lengths are to be compared.

Equation (2) represents, theoretically, a straight line in coordinates $S_e - n$. At n=1 the expected LSC sum has the value of

$$S_e = 2\left(1-\frac{1}{L}\right)\sum_{i=1}^{Z} M_i \frac{L-M_i}{L-1}\,. \tag{4}$$

and theoretically it drops to zero at $n = L$. In fact, though, the $S_e - n$ curve is not exactly a straight line, because the truncated length L of a text (which is part of the equation in question) is obtained by casting off the last, incomplete cell. If the total text's length N is divisible by n, there is no incomplete cell at the text's end, and $L=N$. If, though, N is not divisible by n, the last, incomplete cell, whose size may vary between 0 and $n - 1$, is cast off, so that the truncated text's length L may vary, depending on the values of N and n, between $L = N$ and $L = N - (n - 1)$. As a result, the actual $S_e - n$ curve consists of small steps rather than being an exact straight line, as equation (2) implies. Fortunately, the steps on the $S_e - n$ curve are small (except for very large n) and do not mask the overall linear dependence of S on n, as theoretically predicted.

Let us write the theoretical equation for the expected LSC density ($d_e = S_e / n$) in the following form:

$$d_t = d_e + T = \frac{Q}{n}\,, \tag{5}$$

where d_e is expressed by equation (3) and the constants T and Q are as follows:

$$T = \frac{2}{L}\sum_{i=1}^{Z} M_i \frac{L-M_i}{L-1}, \tag{6}$$

$$Q = 2\sum_{i=1}^{Z} M_i \frac{L-M_i}{L-1}. \tag{7}$$

Equation (5) represents the theoretical hyperbolic function. In logarithmic coordinates, the corresponding theoretical curve is a straight line. However, because the truncation of the text's length, described above, varies for different values of n, the actual curve deviates from the theoretical straight line. To account for that deviation, equation (5) can be modified as follows :

$$d_e = d_t - T = Q\frac{1}{n^q} - T, \tag{8}$$

where for the theoretical function the exponent $q = 1$, but for the actual experimental "curve" it is slightly different from $q = 1$.

All the equations (2) through (7) have been derived for a hypothetical *randomized* text in which the total number N of letters as well as the numbers of appearances of each letter in the text equal these numbers in the original meaningful text. However, for the original meaningful text itself a theoretical calculation of the LSC sums, LSC densities, and specific LSC sums is impossible, because the intrinsic structure of such a text is yet unknown. These quantities have to be found experimentally.

The LSC data for meaningful texts have been obtained by applying a computer program which counted the total number N of letters in the text, as well as M_i – the numbers of occurrences of each letter in the text, divided the texts into k cells each of length n, cast off the incomplete cell if such happened to appear at the text's end, thus truncating the text's length to L, and finally calculated the measured LSC sum S_m, according to equation (1). This operation was repeated for a series of values of n, the cell's size. The described operation produced a set of values of S_m as a function of n. The program had also computed, using eq. (2), the expected LSC sum, S_e, for the same set of values of n.

More than 90 letter strings have been studied, including natural meaningful texts in various languages (Aramaic, Hebrew, Latin, Greek, English, Russian, German, Spanish, Italian, Czech, Finnish, and Yiddish). The LSC data displayed distinctive statistical features, qualitatively identical for all meaningful texts, regardless of language, topic, style, or authorship. These features were, however, absent in meaningless texts, either in artificially constructed, highly ordered ones, or in strings of gibberish randomized in various ways.

3 Experimental Data

The lengths of the studied texts varied from about 5,000 letters to over two million letters. The studied texts included 13 books of the Bible in Hebrew, translations of the Book of Genesis *into all the listed languages* except Yiddish, the entire text of the Torah (the Pentateuch) both in Hebrew and in Aramaic, the Book of Isaiah in Italian, the entire text of the Talmud (which is partly in Hebrew and partly in Aramaic), translations of a part of Tolstoy's novel *War and Peace* into Hebrew and English, the entire text of Melville's novel *Moby Dick* in English, the United Nation's Sea Trade Treaty in English, Shakespeare's

Macbeth in English, Longfellow's *Song of Hiawatha* in English, collections of short (published) stories by the author of this article, one set in English and the other in Russian, and the full text of one issue (October 16, 1988) of the newspaper *Argumenty i Fakty* ("Arguments and Facts") in Russian. Besides the listed original texts, LSC measurements were also conducted on the same texts from which either all vowels or all consonants were removed. Furthermore, experiments were conducted with various artificially constructed texts. Among these artificial texts were highly ordered texts with precisely known structures, for which the LSC sums could be exactly calculated and the results of calculations could be compared with the experimentally measured quantities, thus testing the understanding both of the outcomes of measurements and of the texts' structure.

Also among the studied texts were strings with various degrees of randomness. Some of them were obtained by computer permutations of various elements (paragraphs, verses, words, letters, etc.) of meaningful texts. Other randomized texts were the results of a deliberate effort to artificially create random gibberish from scratch.

Finally, LSC statistics was applied to the yet undeciphered medieval text known as the Voynich manuscript, written in an unknown language and an unknown alphabet. The results of this study are not reported in this paper for two reasons. First, the scope of the obtained data was so large that it would require a separate paper of an even larger size than this one, and that material is more of a cryptological than of a linguistic interest. Secondly, while the results of the study of the Voynich manuscript by the LSC technique seemed to be of great interest, as they shed light on many hitherto unknown characteristics of the manuscript, they had not led to deciphering that mysterious text.

We can generalize the main results of our study as the following two statements:

1. The behavior of the Letter Serial Correlation sums displays certain systematic features, common for all studied texts, regardless of the language, topic, gist, authorship, or style. These features, in particular, distinguish semantically meaningful texts from meaningless strings of characters (thus usually enabling one to determine whether a text is meaningful or gibberish even if its language and/or the meanings of the alphabetical symbols are unknown).

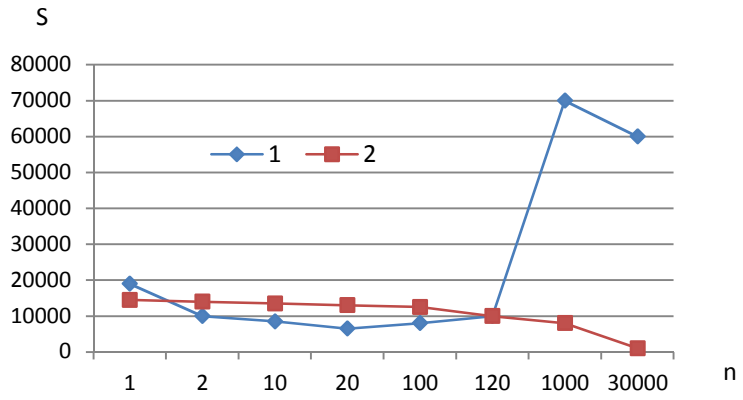


Fig. 1. Measured (S_m, "curve" 1) and expected (S_e, "curve" 2) LSC sums as functions of the cell's size n for the text of the Book of Genesis in Hebrew. The text's length is 78064 letters.

2. There are quantitative differences between the parameters of the LSC statistics for various languages, topics, authorships, etc.

In Fig. 1 the data for the expected (S_e) and measured (S_m) LSC sums are shown for the Hebrew text of "Bereshit" (the Book of Genesis). They exemplify the typical shape of such curves for all the studied meaningful texts (texts in Finnish appear to be an exception which, however, was in fact predicted, as will be discussed later).

When reviewing plots like that exemplified by Fig. 1, it should be realized that the scale for the cell size n on the horizontal axis has deliberately been made non-uniform in order to accommodate the data for the entire range of n in one graph. As n increases, the segments of the n-axis representing the same increase of n become shorter. This leads to the increased curving of the $S_m - n$ and $S_e - n$ graphs toward the n-axis. Were the scale on the n-axis proportional, the $S_e - n$ graph would very closely follow a straight line, according to the theoretical equation (2) while the $S_m - n$ graph would preserve the overall shape shown in Fig. 1 but stretch more to the right. It should be noted that in all figures the values of n, the cell's size, expressed as the numbers of letters in a cell, are integers, as the number of letters cannot be fractional. Hence, the segments of "curves" between the experimental points are drawn only to facilitate the revelation of trends, while by themselves they have no physical meaning.

The LSC "curves" for meaningful texts, regardless of language, alphabet, or the particular semantic contents, all reveal several characteristic points which are as follows:

At small values of n (typically at $n < 3$) the measured LSC sum is usually larger than the expected LSC sum: $S_m > S_e$. As n increases, both the expected and the measured LSC sums decrease, but S_m decreases faster than S_e, so that at some point (to be referred to as Downcross point, DCP, which in Fig 1 is between $n = 1$ and $n = 2$) the curve for S_m crosses the S_e curve and S_m becomes smaller than S_e. If we continue increasing n, both S_m and S_e also continue decreasing until S_m reaches a minimal value at some point $n = n^*$ (to be referred to as the *Minimum Point*, MP) which in Fig. 1 is at $n^* \approx 20$. At $n > n^*$, the expected LSC sum S_e continues its gradual decrease, according to the theoretical equation (2). However, for n exceeding n^*, the measured LSC sum S_m starts increasing. At some point (to be referred to as the *Upcross Point*, UCP) the now ascending S_m curve again crosses the still descending S_e curve. In Fig 1 it happens at $n \approx 120$. If n is increased further, the S_m curve usually reaches a maximum at some point (to be referred as the *Peak Point*, PP). In Fig. 1 it happens at $n \approx 3000$. For even larger n, S_m drops down. The DCP is absent in Finnish (and presumably in Estonian) texts.

While the "curves" for the measured LSC sums are qualitatively identical for all studied languages and types of texts, there are quantitative differences between them. First, the characteristic points DCP, MP, UCP, and PP appear at different values of n, depending on the texts. Second, the *depth of the S_m minimum* at n^* is different for various languages and particular texts.

The variations in the values of n where the DCP point is observed are small; for all the studied texts this point occurs between $n = 1$ and $n = 3$ (except for Finnish and presumably Estonian texts, where DCP is absent). The variations, depending on the language or a specific text, of n^*, at which the MP is observed are more substantial. In all Hebrew and Aramaic texts the MP was observed between $n^* = 21$ and $n^* = 24$. In European languages (Latin, Greek, English, German, Spanish, Italian, Russian, Czech, Yiddish, and Finnish) the MP was observed, depending on the specific text, between $n^* = 30$ and $n^* = 85$. If we also include the texts obtained by eliminating either all vowels or all consonants, the position of the MP happens between $n^* = 8$ and $n^* = 85$.

It seems interesting to report that in many (but not all) cases the value of n^* was found to be close to Z, the number of letters in a given alphabet. For example, in all Hebrew and Aramaic texts studied the MP

was found between $n^* = 21$ and $n^* = 24$ (about 20 texts studied). The alphabets of these two languages each consists of 22 letters. In Czech texts the MP was found at about $n^* = 40$ (the Czech alphabet consists of 41 letters). When all vowels were removed from a Czech text, the location of MP shifted to about $n^* = 28$, which is the number of consonants in the Czech alphabet. In texts of many European languages the MP occurs at n^* between about 25 and 35 (while the sizes of their alphabets are close to these numbers as well). The removal of vowels shifts the position of the MP toward lower values, which, again, are close to the numbers of consonants in these alphabets.

On the other hand, in some other cases MP was found at n^* considerably larger than the size Z of the alphabet. For example, the Minimum Point for the English text of the UN Sea Treaty was found at $n^* = 85$, which is substantially larger than the size ($Z = 26$) of the English alphabet. In a few other texts in European languages n^* was found to be between about 50 and about 70, which also is well above the corresponding alphabets' sizes. Moreover, the units in the equation for S_m are not individual cells, but pairs of cells, so the minima on S_m graphs correspond to the values of $2n^*$ rather than n^*. Therefore, while the alphabet's size has an obvious effect (the longer the alphabet, the higher n^* is expected to be) it seems reasonable to consider the *coincidence* of n^* and the alphabet's size for some of the studied texts as probably accidental. The nature of n^* will be interpreted in the discussion section.

The location of the UCP in all Hebrew and Aramaic texts was found close to $n \approx 150$. In texts written in European languages the UCP was found between about $n \approx 400$ and $n \approx 600$. Of all the characteristic points, UCP is the least informative because it reflects little if any of the intrinsic properties of the studied text. Indeed, this point is where two curves, one for the meaningful text under investigation and the other for a hypothetical randomized text, intersect. While the shape of the S_m curve is determined by the text's structure, it has no relation to the S_e curve, which is for the artificial randomized text, so the structure of the studied text has only a remote bearing on where S_m will accidentally cross the independent S_e curve.

Finally, the Peak Point was observed between $n \approx 3{,}000$ and $n \approx 10{,}000$. As a rule, none of the clearly distinguished characteristic point (DCP, MP, UCP, or PP) was observed on the LSC sums' curves for meaningless strings of letters, so the appearance of these points may serve as an indicator of the semantic meaningfulness of a text.

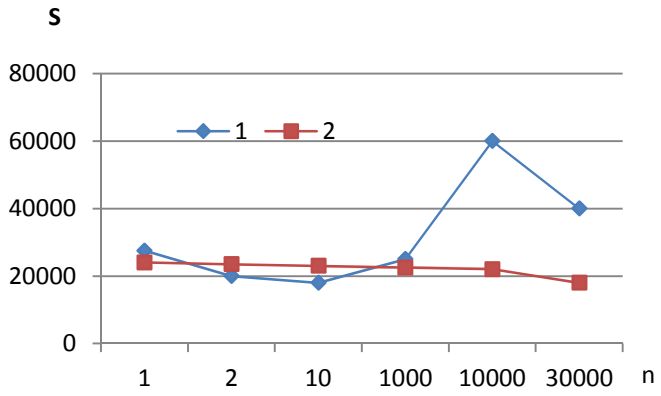


Fig. 2. The measured (S_m, "curve" 1) and expected (S_e, "curve" 2) LSC sums for the set of short stories in Russian. The text's total length is 37000 letters.

For example, in Fig. 2 the expected and measured LSC sums are shown for a text of a set of short stories by the author, published in Russian. We see that despite the drastic difference between the languages (in Fig. 1 it was Hebrew while in Fig. 2 it was Russian), the different text lengths, and the thousands of years between the times of creation of the texts in these two cases, both figures display identical features in regard to the behavior of the variability of letters distribution along the texts.

In both Fig. 1 and Fig. 2, we see the same characteristic point DCP, MP, UCP, and PP, albeit they happen at different values of the cell's size n. A similar picture, with the distinctive points (DCP, MP, UCP, and PP) was observed for *all meaningful texts* in all studied languages (except for Finnish and presumably Estonian, where DCP is absent).

What about randomized texts? Look at Fig. 3, where both expected and measured LSC data are shown for a text obtained via a computer-performed permutation of the letters of the Hebrew text of Genesis. Comparing Fig. 1 with Fig. 3 shows that permutation of letters has completely destroyed the regularities observed in the original meaningful text.

Hence the LSC test allows for an immediate recognition of whether the text is meaningful in some (even completely unknown) language written in any (including the completely unfamiliar) alphabet, or is just a meaningless gibberish.

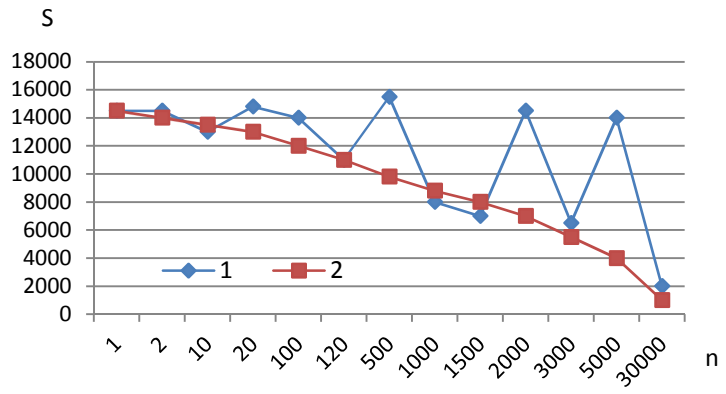


Fig. 3. Measured (S_m, "curve" 1) and expected (S_e, "curve" 2) LSC sums for a text obtained by a random permutation of letters of the Hebrew text of the "Bereshit" (the Book of Genesis). Compare to Fig. 1, where the sums are shown for the same text in its original, non-permuted form.

It should be noted that automatic permutation of the letters of a meaningful text, although converting it into gibberish, does not guarantee its complete randomization. Since the permutation procedure is performed randomly, the number of possible outcomes is very large (it equals $N!$). The overwhelming majority of the permuted strings are meaningless. However, among the vast multitude of the permuted versions of the same original text there is a certain fraction of strings that accidentally contain blocks of letters possessing a certain degree of order, even including segments of a semantically meaningful text.

Therefore we cannot expect the LSC data for a particular permuted string to coincide with the expected LSC sums calculated by equation (2) for a hypothetical randomized text.

Indeed, as we see in Fig 3, the measured LSC sum for this particular permuted version of the text of Genesis is distinct from the expected LSC sum calculated by equation (2) for a hypothetical randomized text of the same length and with the same letter-frequency distribution. At relatively small cell sizes (up to $n \approx 50$) the "curve" of the measured LSC sum is more or less close to the "curve" for the expected sum. This indicates the reasonably high degree of text randomization achieved in this particular permuted string by the letter permutation procedure. At $n>50$ the curve for the measured LSC sum deviates from the curve for the expected LSC sum, the deviations occurring in a

haphazard manner depending on the values of *n*. Similar data have been observed for other versions of the letters strings obtained by random permutations of the same original text. In each permuted version the specific haphazard deviations of S_m from the curve for S_e are of a different shape. The haphazard deviations in question indicate the presence of blocks of letters with a certain degree of order within the overall randomized string, these blocks having different sizes and distribution in each permuted string. If all possible permuted versions of the text in question were available to see, there would be among them also one permutation identical with the expected "curve" S_e vs. *n*, shown in Fig. 1. Moreover, among those permuted texts one will be an exact copy of the original non-permuted text (identity permutation).

What is significant for our study is that the "curves" of the measured LSC sums for randomly permuted texts usually lack those typical features observed for meaningful texts. We don't see on the graphs for randomly permuted strings (Fig. 3) any of the points (DCP, MP, UCP, and PP; see Fig. 1 and Fig. 2) which invariably occur on the LSC graphs for meaningful texts.

Besides the LSC sums, the discrimination between meaningful texts and gibberish can also be done by using the LSC densities. In this case logarithmic coordinates are convenient as the theoretical $log\ d_e - log\ n$ curves for completely randomized strings are straight lines (equations 5, 6, and 7).

Fig. 4 exemplifies the expected and measured LSC density curves (in partially logarithmic coordinates), in this example for the translation of the Book of Genesis into Latin. (For convenience the numbers on the abscissa are given for *n* rather than for *log n*).

Comment. The shape of the "curves" in Fig. 4 is a typical example of a Zipfian law [8] at work. The original Zipf's law stated an empirical functional relation of the word's frequency in a text to *the same word's* "rank" in the order of words' frequencies. Subsequently the term "Zipf's' or Zipfian" law was extended to a wide variety of phenomena; see, for example, [9]. In all of its modifications, Zipfian law always establishes dependence between *two characteristics of the same object*. In the original Zipf's law the *object* was a certain *word*. The two characteristics were the frequency of that word in a text and the "rank" of the same word in the order of frequencies. The data in Fig. 4 present a relation between two quantities—one the cell's size *n* (expressed as the number of letters in the cell) and the other *d*, which is an artificially constructed cumulative property of the entire string.

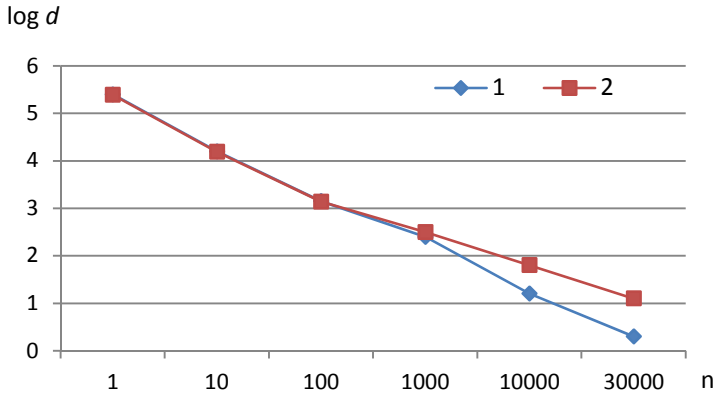


Fig. 4. Logs of the LSC densities: estimated (d_e, "curve" 1) and measured (d_m, "curve" 2) for the translation of the Book of Genesis into Latin.

The cell's size n seems, at a glance, a property of an individual cell, rather than that of the entire string. Were this true, the curves in Fig. 4 would not reflect the relation between two properties of the same object, so the graphs in Fig. 4 would not be the real Zipfian dependencies, but rather look Zipfian-like by accident. In fact, though, as the entire body of this work shows, the cell's size n is a property of the entire string. Indeed, as some value of n is chosen, the string converts into a collection of k equal cells, each of size n. The value of n determines the values of all characteristics relevant to the letter serial correlation analysis.

Moreover, the very value of d is determined by the value of n. Hence, both n and d are properties of the entire string, thus justifying the interpretation of the curves in Fig. 4 as genuine Zipfian dependencies.

The curve for d_{m} in fig. 4 obviously consists of two parts. One part, at $n < n^*$, is practically indistinguishable from the curve for d_e, which is of the expected LSC density. The second part of the curve for d_m, at $n > n^*$, is clearly different from the curve for d_e. Using the least squares fit, we found that the entire curve for d_e as well as both parts of the curve for d_m, are all well approximated by straight lines.

In this particular example, the corresponding equations are as follows: for the expected LSC density, $d_e = 1{,}729{,}189\, n^{-1.021}$ (correlation coefficient is 0.9992); for the measured LSC density at $n < 22$, $d_m = 1{,}788{,}292\, n^{-1.073}$ (correlation coefficient is 0.99992); for $n > 22$, $d_m = 1{,}500{,}610\, n^{-0.732}$ (correlation coefficient is 0.99965).

The negative exponents in the above equations all differ slightly from 1. As discussed earlier, in the case of the expected LSC densities, the deviations of the exponent from the value of 1 (the latter corresponds to the theoretical hyperbolic curve) reflect the effect of the text's truncation when the end cell happens to be incomplete and is cast off. In case of a measured LSC density when the shape of $d_m - n$ function cannot be theoretically calculated, the deviation of the exponents from unity reflects the difference in the letter-variability distribution between meaningful texts and their permuted versions.

From the above data (which exemplify the similar results obtained for a wide variety of texts in 12 languages) it follows that LSC statistics may be considered a reliable tool for discriminating between meaningful texts, regardless of language and alphabet, on the one hand, and gibberish, on the other.

However, we still need to test whether or not meaningless strings (besides those obtained by permutations of letters of meaningful originals) can sometimes masquerade as meaningful texts by producing LSC data imitating those exemplified in Figures 1 and 2.

To this end various versions of meaningless strings, those possessing a high degree of order as well as those which are highly chaotic, were studied. First, the LSC statistics were applied to strings obtained by various methods of permutation of the meaningful original text.

In one version of the procedure, the words within each paragraph of a meaningful original text were randomly permuted by a computer while the paragraphs themselves stayed in their original places. As long as the doubled cell size ($2n$) is not exceeding the average word length, the behavior of LSC sums, as could be expected, remained similar to the one observed for meaningful texts. However, as the doubled cell size ($2n$) becomes larger than the average word length, the LSC sums for the words-within-paragraphs-permuted strings deviate markedly from those for the meaningful texts.

A similar effect was observed in strings obtained by random permutations of the paragraphs of the original meaningful text while the words and letters within the paragraphs remained intact. If paragraphs are short and have been randomly permuted, the overall text becomes in a certain sense meaningless. Since, however, the text within the paragraphs remains intact, each paragraph preserves, within its confines, the structure of a meaningful text.

Therefore, although a string obtained via random permutations of the paragraphs of a meaningful text (keeping the texts within the paragraphs intact) loses its logical consistency and, hence, can be

characterized in a certain sense as meaningless, it could be expected that for the doubled cell sizes not exceeding the average paragraph length the LSC curves for such permuted strings would look similar to the case of a meaningful text. Indeed, such a behavior was observed for the strings obtained by the described version of permutation. To illustrate the described behavior, in Fig. 5 the LSC curves are shown for the Hebrew text of the book of Genesis obtained via the described permutation of verses without modifying the text within the verses.

At $n < 22$, i.e. $2n < 44$, when the doubled cell's size is less than the average size of a verse, the measured LSC sum's curve behaves similarly to the curves for meaningful texts: the Downcross Point and the Minimum point for this permuted string are observed at about the same values on n as for a meaningful text.

However, at $n > n^* = 22$ the measured LSC sum for the text with permuted verses behaves differently from meaningful texts, approaching the behavior of fully randomized texts.

These data indicate that there may be (albeit it seems not very likely) two types of order related to the letter-variability distribution along the text—a short range order and a long range order. Shuffling paragraphs (or verses) destroys the putative long range order but leaves intact the short range order, and the shape of curves for the measured LSC sums might reflect it. (This question will be discussed a little later.)

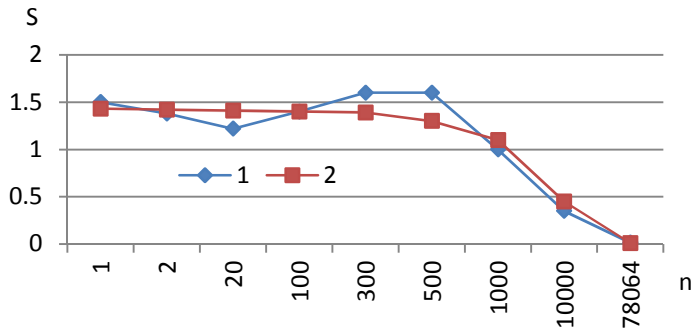


Fig. 5. Measured (S_m, "curve" 1) and expected (S_e, "curve" 2) LSC sums as functions of the cell's size n for a text obtained by a random permutation of verses in the Hebrew text of the Book of Genesis (without permuting letters or words within the verses). The text's length is 78,064 letters. The scale on the abscissa is logarithmic, but for convenience it is marked in the values of n rather than of log n.

In one more version of permutation, all *words* of the text were randomly permuted by the computer without permuting letters within the words. In this case the curve for the measured LSC sum was similar to those for meaningful texts as long as the cell's doubled size 2*n* was less than the average length of a word. However, when 2*n* exceeded the average word length, the measured LSC sum behaved differently from the meaningful original, but similar to the curves for the texts randomized by letters permutations.

In another set of control experiments certain artificially created meaningless strings, some with highly ordered and others with chaotic structures were constructed.

One such text was formed by repeating letters of the English alphabet 3,000 times each (first the letter A was repeated 3,000 times, then the letter B, etc.). This string was 63,000 letters long (it contained no segments for the last five letters of the English alphabet). This string was highly ordered so its entropy was close to zero. Since the structure of that text was precisely known, it was possible to theoretically compute its LSC sum and density. The precise formulae for calculating the measured LSC sums and densities for that text are shown in Appendix 2. While the derivations of these formulae are omitted to keep the paper's size within reasonable limits, the validity of the formulae in question follows from the almost perfect coincidence of the data obtained experimentally and those calculated using these formulae. (Anybody may get the detailed derivation of the formula in question by requesting it from the author.) In Fig. 6 the plot of the LSC density vs. cell size (in log-log coordinates) is shown for the near-zero-entropy string in question. The results of measurements and calculations (conducted for the same set of discrete cell sizes) coincided in this case so closely that the two curves could not be resolved from each other, so the same zigzag-shaped graph in Fig. 6 represents the data for both the measurement and calculation.

This result testifies that we have developed a reasonable understanding of the LSC effect and its relation to the text's structure. As Fig. 6 shows, the behavior of the LSC statistics in the described near-zero-entropy meaningless string has nothing in common with the behavior of the corresponding quantities for meaningful texts (illustrated in Fig. 4).

Another artificial string was formed by sequentially repeating the English alphabet 2422 times. The entropy of that meaningless string is a little higher than for the previously discussed low entropy texts, but it

is still very low. For this text the shape of the $S_m - n$ curve also turned out to be different from the curves for meaningful texts.

One more artificial meaningless string of low entropy was created by first repeating the first half of the English alphabet, i.e. the string ABCDEFGHIJKLM, 17 times; to its end a string was concatenated which consisted of the letters BCDEGFHIJKLMN repeated 17 times; then the letters CDEFGHIJKLMNO, repeated 17 times, followed, etc., until the last substring comprising the second half of the alphabet, also repeated 17 times, completed the text.

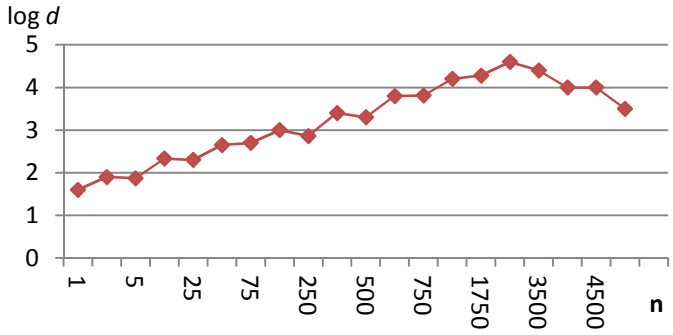


Fig. 6. Dependence of both measured (d_m) and calculated (d_c) Letter Serial Correlation densities on cell size n (in log-log coordinates) for an artificially created highly ordered string 63000 letters long. For convenience, the values on the abscissa are indicated for n rather than for logarithms of n. The upper cusp corresponds to $n = m$. To the left of the cusp $n < m$, to the right $n > m$ (in this sample $m = 3000$).

The procedure was repeated 7 times, so the total length of that text was 20111 letters. Again the $S_m - n$ curve for this highly ordered text was distinctively different from the $S_m - n$ curves for meaningful texts.

Finally, one more meaningless text was made up by randomly hitting the keys on the computer keyboard, trying to avoid favoring any particular letters at the expense of other letters. Unlike the previously discussed artificial texts, which all were substantially ordered and thus had low entropy, this string (which was 10,000 letters long) was prepared with the intention of yielding a highly randomized string thus possessing entropy substantially exceeding that of meaningful texts.

It is known [10, 11] that actions of humans cannot be effected in a genuinely random manner. Despite the strenuous effort to avoid any selectivity in hitting the keyboard buttons, a human operator will

subconsciously but inevitably hit the keys in a not fully random way. As expected, the S_m *vs.* n curve for the supposedly random string obtained as described revealed certain subconscious selectivity which resulted in a letter frequency distribution different from a fully random string. To a certain extent the letter frequency distribution in the artificial, supposedly random text, indeed turned out to be more uniform than in meaningful texts. (In a perfectly random text the letter frequency distribution is ideally uniform). However, it was not as uniform as it should be in a perfectly random string. Therefore, the $S_m - n$ curve for this artificial high-entropy text displays certain features resembling the data for meaningful texts (for example, a minimum at a certain value n^* of a cell size). Although these features are not as clearly evident as they are for meaningful texts, they may cause doubts in regard to the distinction between *disordered gibberish* and meaningful text insofar as the LSC statistics is applied. While this phenomenon is perhaps of interest for psychology, in our case we needed to determine whether or not the LSC statistics enables us to distinguish between semantically meaningful strings and disordered gibberish of high entropy.

It was found that the plots of *specific* LSC sums for meaningful texts are more clearly different from those for the artificial high-entropy gibberish than are the plots of S_m sums. Furthermore, the data are distinctively different for meaningful texts and for high-entropy gibberish if a text is divided into halves and the LSC statistics are compared for both halves. In the case of a meaningful text, the exact locations of the MP (i.e. the value of n^*) as well as the "depth" of the minimum typically are different for the two halves of the text. On the other hand, in the case of artificial high-entropy gibberish the characteristic points for both halves of the text are almost identical.

As mentioned before, we have also studied texts obtained by removing either all vowels or all consonants from the meaningful texts. These studies have revealed that the "shrunk" texts composed of either only consonants or only vowels preserve all the features of the LSC statistics observed for the original, full versions of the same texts. On the curves of the measured LSC sums for "shrunk" texts all characteristic points DCP, MP, UCP, And PP, discussed earlier, are clearly seen, as they are on the curves for the full, all-letters versions.

(There is a quantitative difference between the LSC sum curves for the full, all-letters versions, and for the "shrunk" only-vowels or only-consonants versions. The removal of all vowels, and even more of all consonants, causes a shift of the MP to lower values of n^* and also

decreases the "depth of minimum" on those curves.) This points to the deeply intrinsic character of the LSC statistics' behavior, which is not destroyed even by such a brutal mutilation of texts as the removal of all vowels or of all consonants.

4 Discussion

A detailed discussion of the entirety of our LSC data (which comprise over 300 graphs and scores of tables) cannot be done within the confines of a reasonable paper size. Therefore only a brief discussion of the most salient points will be offered here.

First, we will discuss the nature of the Downcross Point (DCP) observed on measured LSC curves for meaningful texts. The explanation in this case seems to be almost obvious. Recall that the DCP was always observed at the cell's size between $n = 1$ and $n = 3$. In other words, at $n = 1$, i.e. when the cells contain only one letter each, the measured LSC sum S_m for meaningful texts is slightly larger than the expected LSC sum S_e , calculated for a text obtained by permutation of letters of the original meaningful text. This, of course, is expected. Indeed, at $n = 1$ each cell holds just one letter.

Since the terms in the LSC sum are contributed by *pairs* of neighboring cells, there are only two possibilities. If both neighboring cells of size $n = 1$ happen to contain the same letter, the term contributed to the LSC sum by that pair of cells equals zero. If, though, the neighboring cells of that size contain different letters, such pair of cells contributes to the LSC sum a term of 2 (since each of the differing letters in question contributes 1 to the sum; see eq. (1)).

It is easy to figure out that the maximum value of S_m (for $n = 1$) is observed when no pair of neighboring cells contains the same letter in both cells; the sum is in this case $S_m = 2(L - 1)$ where in this case $L = k$. Therefore, the more pairs of neighboring cells of size $n = 1$ hold the same letter, the smaller the LSC sum is. In natural texts doubling of letters is rare; the probability of any pair of neighboring cells of size $n = 1$ containing the same letter is less than the probability of them holding different letters.

On the other hand, in a randomized texts all letters are almost equally likely to occur in any cell (except of the effect of the letters various frequency, mentioned above), so if in cell j there is letter x, the probability of the same letter x also appearing in cell $(j+1)$ is almost the same as for any other letter, say y, of the alphabet. (Strictly speaking,

this assertion is exactly valid only for a perfectly random text, while the calculation of the expected sum was conducted for texts randomized by letter permutations of the original meaningful text; however, the calculation has shown that for not very short texts the quantitative difference between the values of the expected LSC sum for perfectly random texts and for letter-permuted texts is, in practical terms, utterly negligible; therefore the above assertion remains practically valid for our data).

As a result, a randomized text at n=1 usually contains more pairs of neighboring cells with the same letter in each than the original meaningful text. Hence the LSC sum for a randomized text at $n = 1$ includes more terms equal to zero than the corresponding sum for a meaningful text. This results in a slightly larger S_m at $n = 1$ for meaningful texts than for randomized strings. At $n > 1$, when a cell contains more than one letter, the LSC sums, both expected and measured, decrease. Indeed, if cells contain only 1 letter each, each time two neighboring cells hold different letters it means a 100% change of a cell's content from cell to cell.

If, though, cells contain more than 1 letter each, only a fraction of neighboring cells will have the entire set of letters in each cell different from its neighbor; some other pairs of cells will have only partially different contents, so the change of a cell's content from cell to cell, on the average, will be less than 100% (i.e. the relative letter variability decreases for $n > 1$).

The LSC sum is larger when the variability of letters distribution is larger. Since for $n > 1$ the relative variability decreases, the LSC sums drops. It drops faster for meaningful texts than for randomized ones because in the latter this effect is mitigated by the much larger degree of the overall randomness of the letters distribution. As a result, the descending curve for the gradually decreasing S_m crosses at the DCP the curve for the also decreasing, but at a slower pace, S_e.

If the above explanation is correct, certain predictions can be suggested. If a certain language's orthography requires a frequent doubling of identical letters, for a meaningful text in such a language the measured LSC sum will contain, at $n = 1$, a slightly larger fraction of pairs of neighboring cells both holding the same letter. Such pairs of cells will contribute to the LSC sum terms equal to zero, and this will result in a decrease of S_m for such a text, making it less than the expected LSC sum S_e at n=1. Finnish and Estonian orthography require a frequent doubling of both consonants and vowels. Therefore, based on the above interpretation of the Downcross Point, it could be

predicted that for Finnish (and presumably Estonian) texts the measured LSC sum S_m at $n = 1$ would be no larger than the expected LSC sum S_e, as was observed in the variety of other texts, but, on the contrary, would be below the expected LSC sum. This prediction has been fully confirmed experimentally for Finnish texts.

Based on these data one more prediction was made. Italian orthography requires a frequent doubling of consonants but not of vowels. Therefore, for regular meaningful Italian texts no "abnormality" in the mutual location of S_m and S_e curves at $n = 1$ can be expected. Indeed, the LSC curves for Italian texts had the usual configuration wherein at $n = 1$ the measured LSC sum is slightly larger than the expected LSC sum. It could be expected, though, that in Italian texts stripped of all vowels the frequent doubling of consonants would result in the inversion of the S_m and S_e curves at $n = 1$, as was observed for Finnish texts. This expectation was also fulfilled.

The described observations favor our interpretation of the Downcross Point.

Let us now discuss the Minimum Point. The value of the measured LSC sum S_m is determined by the *variability* of the letter distribution along the text. Recall that the terms in the S_m sum are calculated for *pairs of adjacent cells.* The more identical letters happen to occur *on the average* within the length of $2n$, the less S_m is. Obviously, then, the minimum on the $S_m - n$ curve must occur at such cell's size n^*, which corresponds to the *minimal average variability* of the letters distribution within a segment of the size $2n^*$.

The observation of the MP means the revelation of what can be referred to as an *average Domain of Minimal Letter Variability* (DMLV) whose size is $2n^*$ and which exists in all meaningful texts using an alphabetical writing system, regardless of language, style, authorship, alphabet, etc.

While the DMLV is consistently present in all meaningful texts, it is usually absent in gibberish, both of the highly ordered and the highly randomized kinds. (Although in extremely rare cases a string of gibberish may accidentally happen to have a DMLV, this would be an exceptional occurrence, while in meaningful texts it is a rule.)

The statement asserting the consistent existence of a DMLV in all meaningful texts (but its usual absence in gibberish) follows directly from the observation of a distinctive minimum on the $S_m - n$ curves, i.e. it is simply a statement of fact. Its interpretation, although post-factum, does not seem very difficult.

It seems reasonable to postulate that the size of a DMLV is related to the size of a text's segment wherein a certain topic is covered. Then it can be expected that certain words related to that topic occur within that segment more often than on average in the text as a whole. Consequently, a certain set of letters is also expected to occur within that text's segment more often than in the rest of the text. This means a lower letter variability within the segment in question, which contributes to a smaller value of S_m. The size of a DMLV may be expected to be connected to the *average size* of a text's segments covering individual topics.

Our interpretation jibes well with the observed variations between the positions of MP in various texts. For example, the Hebrew and Aramaic texts are written in alphabets each containing only consonants, with the total of 22 letters in the alphabet. On the other hand, the most common European languages use substantially longer alphabets (for example, the English alphabet has 26 letters; the Russian alphabet has 33 letters, while the Czech alphabet has 41 letters). These variations alone necessarily must affect the size of a text's segment covering a certain topic. However, besides the alphabet's size, the peculiar ways in which each language structures sentences enhances the variations in the DMLV. Here is a simple illustration. Consider a maxim that came from the ancient Hebrew texts but has become part of many ancient and modern languages. Let us write that maxim in several languages. Start with its original form in Hebrew, which looks like אין נביא בעיר (to be read from right to left). Transliterated into Latin characters, it takes the following form: *EIN NVI BIRO*. Its length is only 10 letters.

Now let us write the conventional translations of that maxim into English, German, Russian, and Ukrainian. English: *There is no prophet in one's native town.* (31 letters, of which 19 are consonants). German: *Es gibt kein Prophet in seiner Stadt.* (30 letters, of which 19 are consonants). Russian (rendered in Latin letters): *Net proroka v otechestve svoem* (25 letters, of which 15 are consonants; the combination *ch* in the Russian alphabet is rendered by one letter). Ukrainian: (rendered in Latin letters): *Nema proroka u ridnomu misti*. (25 letters, of which 14 are consonants).

Obviously, the Hebrew text requires substantially fewer letters to cover a certain topic, so the DMLV for Hebrew naturally is shorter, than, say, for English or Russian, and the minimum on the S_m "curve" for Hebrew texts appears at lower n (usually about 20–24) than, say, for English texts (typically somewhere about 70 and even more). The unusually large $n^* = 85$ for the UN Sea Trade Treaty also can be

interpreted on the same basis: it is written in a heavy legalese; such documents are known for a pedantic verbosity, wherein each statement is expressed with multiple asides and additional clauses, which makes the segment of a text, covering a certain topic, substantially longer than in non-legal texts. This shifts n^* to larger values than in non-legalese-written texts.

A natural unit of a semantic content is a sentence. Therefore it may be surmised that the size of a DMLV is somehow related to the average length of a sentence. It hardly could be the length of one sentence, because if $2n^*$ were about one sentence long, n^* would be about half a sentence long, and in this case to ensure the minimal letters' variability, two halves of one sentence would need to contain, on the average, almost the same set of letters, which can hardly be expected. Therefore it seems reasonable to expect that DMLV should comprise several sentences, albeit not too many, so that the set of sentences within the scope of an average DMLV covers a specific narrow subject.

To test that hypothesis we have measured the average lengths of sentences in a variety of texts and compared them with the values of $2n^*$ for these texts.

The value of $2n^*$ varies, depending on languages and specific texts, and usually is between 40 and 170 letters. On the other hand, the average length of a sentence, depending on texts, was found to be between $0.4n^*$ and $1.35n^*$, the mean value being about $0.8n^*$. Therefore it can be stated that there is in all meaningful texts an average Domain of Minimal Letter Variability which is between 1.5 and 4.5 sentences long, its average length for a variety of texts being about 2.5 sentences. Apparently that is the average length of a text's segment typically covering individual subjects and hence containing a limited variety of letters. As the text's segment becomes longer than, on the average, the length of the DMLV, the subject changes, and with it also the words used, and hence the letter composition becomes more varied, so the measured LSC sum S_m increases above the minimum.

Finally, let us discuss the peak (PP) on the $S_m - n$ curves for meaningful texts. To decipher the nature of that peak special tests have been conducted, in which two types of texts were compared. To this end a long text would be chosen, for example the text of several sequential chapters of Tolstoy's novel *War and Peace* in English translation.

The length of the text subjected to the test in one particular case was 180,000 letters. This text was then divided into 18 equal segments of 10,000 letters. The LSC sum was measured for the first segment. Then

the text was gradually enlarged by sequentially concatenating additional segments of the same size. The LSC sums were measured at each step of the text's gradual enlargement. In one set of tests, at each step the added segment was *different* from the previously concatenated one, being the *next* segment in the sequence constituting the 180,000 letter-long original text. In another set of tests, at each step *the same* initial segment was repeatedly concatenated to the string, until the total text comprised 18 identical parts each 10,000 letters long. This way a strong long range order was artificially generated in the tested text, while in the first set of tests the long range order, if such existed, was limited to that existing in the text naturally.

Comparing the two described types of a text, it was found that the LSC sums behave quite differently in the two texts in question. These data indicated that the natural meaningful texts possess no long range order. As the cell size n increases, each cell encompasses a larger chunk of a text. As the length of the text within a cell increases, local violations of order accumulate, until no order can be observed any longer. Since the *short range* order naturally does not exist anymore for such large values of n, and the long range does not exist in natural meaningful texts anyway, for such large n the text starts behaving similar to a randomized one. For the latter, as the behavior of the expected sum S_e shows, the LSC sum always decreases with the increase of n. Hence the $S_m - n$ curve, which is ascending at smaller n, now changes to a descending one, typical of randomized texts. Therefore the peak on the $S_m - n$ curve corresponds to such cell sizes where the LSC type of order in the text completely disappears, and the LSC "curve" follows the behavior typical of random texts. .

5 Conclusion

As the data presented here show, the LSC statistics makes it possible in many cases to reliably distinguish semantically meaningful texts from gibberish, regardless of the alphabet in use, language, style, authorship, etc. The LSC statistics have revealed certain hidden features of the order intrinsic in meaningful texts, as, for example, the existence in all such texts of an average Domain of Minimal Letter Variability. Furtermore, a conection was revealed between the LSC statistics and Zipf's law.

The ancient Hebrew and Aramaic texts display exactly the same behavior regarding the letters' variability distribution along the text as the text of a Russian newspaper printed in 1988, or as a Shakespeare's play in English, or as a translation of Genesis into Czech. Languages differ in their vocabulary, grammar, idioms, and alphabets, but somewhere on a deeper level they all seem to follow the same statistical features, which perhaps points to their common origin from a single source.

ACKNOWLEDGEMENTS. Dr. Brendan McKay of the Australia National University, Canberra, Australia, at various periods of time participated in this work, including the preliminary discussion of the idea of LSC, writing the computer programs, used for performing the computation of the LSC data, and running various texts through these programs. The author deeply appreciates Dr. McKay's contribution. The author is also thankful to anonimous reviewers for pithy comments which served to the paper's substantial mprovement.

Appendix 1. Derivation of the Formula for Expected Letter Serial Correlation Sum

Recall that $X_{i,j}$ denotes the number of occurrences of letter x_i in a cell number j. Since all cells are of the same length n, we have

$$\mathrm{Var}\,(X_{i,j}) = \mathrm{Var}\,(X_{i,\,j+1}), \tag{A1}$$

$$\mathrm{E}(X_{i,j}) = \mathrm{E}\,(X_{i,\,j+1}), \tag{A2}$$

where Var(X) is the variance of X and E is the expected value of X.

Step 1. Variance is calculated [12] as follows :

$$\mathrm{Var}\,(X) = \mathrm{E}\,(X^2) - [\mathrm{E}\,(X)]^2, \tag{A3}$$

where the first term is the expected square of X and the second term is the square of the expected X.

Consider now the expression $\mathrm{E}[(X_{i,j} + X_{i,\,j+1})]^2$ which is the expected square of the sum of the values of X in two sequential cells. From equation (A3) we obtain

$$\mathrm{E}[(X_{i,j} + X_{i,\,j+1})^2] = \mathrm{Var}\,(X_{i,j} + X_{i,j+1}) + [\mathrm{E}(X_{i,j} + X_{i,j+1})]^2. \tag{A4}$$

The expected value of a sum equals the sum of the expected values of the items it comprises [8]. Then, accounting for equation (A2), we obtain from equation (A4):

$$E[(X_{i,j} + X_{i,\,j+1})^{\,2}] = \mathrm{Var}\,(X_{i,j} + X_{i,\,j+1}) + 4[E(X_{i,j})]^2. \qquad \text{(A5)}$$

Now consider the expression

$$E[(X_{i,j} - X_{i,\,j+1})^2] + E[(X_{i,j} + X_{i,\,j+1})^2. \qquad \text{(A6)}$$

Replacing the expected value of a sum with the sum of expected values of its constituent items and accounting for (A2), we obtain from (A6) the following set of algebraic transformations:

$$\begin{aligned} &E[(X_{i,j} - X_{i,\,j+1})^2] + E[(X_{i,j} + X_{i,\,j+1})^{\,2}] = \\ &E[(X_{i,j} - X_{i,\,j+1})^2 + (X_{i,j} + X_{i,\,j+1})^{\,2}] = \\ &E[(X^2_{i,j} + X^2_{i,j+1} - 2X_{i,j}\,X_{i,j+1} + X^2_{i,j} + X^2_{i,j+1} + 2\,X_{i,j}\,X_{i,\,j+1}] = \\ &E[2\,X^2_{i,j} + 2\,X^2_{i,\,j+1}] = E\,[4\,X^2_{i,j}] = 4E[X^2_{i,j}] \end{aligned} \qquad \text{(A7)}$$

Now subtract (A5) from (A7):

$$[(X_{i,j} - X_{i,\,j+1})^2] = 4\,E\,[X^2_{i,j}] - 4\,[E(X_{i,j})]^2 - \mathrm{Var}\,(X_{i,j} + X_{i,\,j+1}) \qquad \text{(A8)}$$

From equation (A3) we can see that the first two terms in the right side of equation (A8) equal $4\mathrm{Var}\,(X_{i,j})$. Then

$$E[(X_{i,j} - X_{i,\,j+1})^2] = 4\ \mathrm{Var}\,(X_{i,j}) - \mathrm{Var}\,(X_{i,j} + X_{i,\,j+1}) \qquad \text{(A9)}$$

Comment. If we dealt with perfectly random texts, $X_{i,j}$ and $X_{i,j+1}$ would be independent random variables. However, we are deriving formulas for a text randomized by a permutation of the letters of an original meaningful text, so the permuted text is not perfectly random. Unlike for a perfectly random text, the stock of available letters in our case is limited to those letters present in the original meaningful text and in the same numbers. Therefore if a certain letter *x* occurs in a cell, this decreases the stock of this letter available for the next cell and thus diminishes the probability of *x's* occurrence in the next cell. Hence there is a certain negative correlation between $X_{i,j}$ and $X_{i,j+1}$ which therefore are not independent variables. In such cases the variance of a sum cannot be replaced with the sum of variances of its constituent items so the variances of both $X_{i,j}$ and $(X_{i,j} + X_{i,\,j+1})$ must be calculated and inserted into equation (A9) separately. If, though, variables $X_{i,j}$ and $X_{i,\,j+1}$ were independent, the right side of equation (A9) would equal $2\mathrm{Var}\,(X_{i,j})$.

Step 2. We have to choose now the distribution function for the quantity $X_{i,j}$ within a cell. Our options are limited to the choice between the multinomial and hypergeometric distributions [13]. The multinomial distribution is applicable to tests with replacement while the hypergeometric distribution is applicable to tests without replacement. In our case, if a letter, say *x*, occurs in a cell once, this decreases the probability it will occur again in the same (or the next) cell, because the stock of letters is limited to those actually found in the original meaningful string. Therefore the conditions under which our calculation is conducted meet the definition of tests without replacement. In other words, we postulate the hypergeometric distribution of letters' frequencies within the cells. (While this choice is theoretically well justified, it has a very little significance in practical terms. As the pertinent calculation shows, the final formulae of S_e differ between the cases of a hypergeometric and a multinomial distributions only by the factor of $L / (L - 1)$, where L is the truncated (if need be) length of the text, expressed as the number of letters. Obviously, except for extremely short strings (which are hardly of interest) the above factor is so close to unity that the difference between the formulae for the two listed distributions is utterly negligible; for the sake of theoretical purity we calculate here the expected LSC sum for a hypergeometric distribution.)

For the hypergeometric distribution, the variance is calculated as [12]:

$$\text{Var}\,(X_{i,j}) = (L - m)\, mp\,(1 - \text{p}) / (L - 1), \qquad \text{(A10)}$$

where m is the sampling size and $p = M_i / L$. Recall that M_i is the number of occurrences of the letter x_i in the entire string and L is the truncated (if need be) length of the text expressed as a number of letters.

For the first term in the right side of equation (A9) the sampling size m equals the cell size: $m = n = L / k$. For the second term on the right side of (A9) the sampling size is twice as large: $m = 2n = 2L / k$. Then we can write for the first term on the right side of (A9):

$$4\,\text{Var}\,(X_{i,j}) = 4(L - L / k)\,(1 - M_i / L)\, M_i / k\,(L - 1),$$

or, after a simple algebraic transformation

$$4\,\text{Var}\,(X_{i,j}) = 4\, M_i\,(L - M_i)\,(1 - 1 / k) / k\,(L - 1). \qquad \text{(A11)}$$

Similarly, for the second term on the right side of (A9) with its doubled sampling size we obtain

$$\mathrm{Var}\,(X_{i,j} + X_{i,\,j+1}) = 2\,(1 - 2 / k)\,M_i\,(L - M_i) / k\,(L - 1). \quad \text{(A11a)}$$

Finally, plugging (A11) and (A11a) into (A9), we find

$$\mathrm{E}[(X_{i,j} - X_{i,\,j+1})^2] = 2\,M_i\,(L - M_i) / k\,(L - 1). \quad \text{(A12)}$$

To complete our calculation, we have to sum (A12) over all letters of the alphabet (from $i = 1$ to $i = Z$) and over all pairs of neighboring cells (from $j = 1$ to $j = \mathrm{k} - 1$). Since, however, all cells are of the same size, the summation over j can be replaced with a multiplication by the value of $k - 1$. This results in the formula

$$S_e = 2\left(1 - \frac{1}{k}\right)\sum_{i=1}^{Z} M_i \frac{L - M_i}{L - 1}.$$

Equation (2) in the body of the text is a replica of the above equation with one modification: the number k of cells which appears in the above equation, is replaced in equation (2) with its expression through the cell size n and the string's truncated length ($k = L / n$).

Appendix 2. Formula of LSC Sum for an Artificial Low Entropy Text Composed of Repeated Letters

Consider a string L letters long composed of Z equal *segments*, each m letters long, where Z is the number of letters in the alphabet. Each m-long segment contains one particular letter, repeated m times. There are no two segments containing the same letter. For example, such a string can have $Z = 26$ segments, of which the first one contains m times the letter A, the second segment m times the letter B, etc., up to the segment number 26 which contains m times the letter Z. As before, we also divide this string into k *cells* each n letters long, so that $kn = mZ = L$. Obviously the boundaries between cells and those between segments generally will not coincide. The value of m is fixed for a particular string while the value of n varies as we calculate (or measure) the LSC sum. Since the structure of this string is precisely known, we can theoretically calculate the LSC sum for that string.

We have to distinguish between two cases, in one $m > n$ and in the other $n > m$. Introduce the following notations:

For $m > n$: $m / n = s + v$;
For $n > m$: $n / m = r + w$,

where *s* and *r* are integer parts while *v* and *w* are fractional parts of the corresponding quotients.

As long as $m > n$, the calculated LSC sum is found from the following equation (its derivation is freely available to anybody who would request it from the author. Its validity is ascertained by the almost perfect coincidence of the data obtained via that equation with the measured data):

$$S_c = 2 j^* n^2 \left[\sum_{i=1}^{i^*} (1 - iv)^2 + \sum_{i=1}^{i^*} (iv)^2 \right].$$

For $n > m$ the formula for the LSC (its derivation is also available on request) becomes

$$S_c = 2 t^* mn \left[\sum_{i=1}^{i^*} (1 - iw)^2 + \sum_{i=1}^{i^*} (iw)^2 \right].$$

In these equations $j^* = (Z - 1) / i^*$ and $t^* = (k - 1)(n - mr) / m$; i^* is either the integer part of the quotient $1 / v$ (for the case of $m > n$) or the integer part of the quotient $1 / w$ (for the case of $n > m$).

In those cases where either m / n (if $m > n$) or n / m (if $n > m$) are integers, the above equations convert into much simpler versions. For $m > n$ in such cases

$$S_c = 2n^2 (Z - 1). \quad \text{(i)}$$

For $n > m$ in such cases

$$S_c = 2mn (k - 1). \quad \text{(ii)}$$

For the particular case of $n = m$ both equations (i) and (ii) yield identical results.

The LSC *density* d_c is obtained from all the quoted formulas via the division by the cell size *n*.

The points between those for the integer values of n, form a zigzag-shaped curve which has no meaning in itself but shows the trends.

The uppermost cusp on the curve in Fig. 6, which separates the ascending and the descending branches of the graph, corresponds to $m = n$. (The particular curve in fig. 6 relates to a text where $m = 3{,}000$, and the total length is 63,000 letters). The results of calculations using the quoted formulas turned out to be very close to the results of a direct measurement of LSC density, so that the calculated and measured curves practically coincided. This observation may serve as

confirmation that we have developed a reasonable understanding of both the structure of texts, insofar as their letters variability distribution is in question, and of the working of the LSC statistics.

References

1. R. S. Pindyck, D. L. Rubenfeld. *Economic Models and Economic Forecasts*. McGrow Hill, NY, 2000.
2. T. J. Bruno & P. D. N. Svoronos. *CRC Handbook of Fundamental Spectroscopic Correlation Charts.* CRC Press, NJ, 2005.
3. J. Tyrangiel. *Why Pop Music Sounds Perfect*. Time Magazine, No. 2, 2009.
4. R. J. Solomonoff. *A Preliminary Report on a General Theory of Inductive Inference*. Cambridge, MA. Report ZTB138. Zator Co., 1960.
5. A. N. Kolmogorov. Three Approaches to the Quantitative Definition of Information. *Problems of Information Transmission*. **1**: 1–17, 1965.
6. G. J. Chaitin. Randomness and Mathematical Proof. *Scientific American*, **5**, 232 –238, 1975.
7. P. Vitanyi, "Meaningful information, http://front.math.ucdavis.edu/ cs.CC/0111053, 2000.
8. C. D. Manning. Foundations of Statistical Natural Language Procesing. MIT Press, Cambridge MA, 1999.
9. N. L. Johnson, S.Kotz and A. W. Kemp. *Univariente Discrete Distribution*, 2nd ed., John Wiley and Son, NY, 1992.
10. M. Bar-Hillel and W. A. Wagenaar. The perception of randomness. *Advances in Applied Mathematics*, **12**, 428–454, 1991.
11. G. Keren and C. Lewis (eds). *A Handbook for Data Analysis in the Behavioral Sciences*. Hillsdale, NJ: Lawrence Erlbaum, 1993.
12. R. J. Larsen and M. L. Marx. *An Introduction to Mathematical Statistics and its Applications*. Englewood, NJ, Prentice-Hall Publishers, 1986.
13. P. Olofsson. *Probability, Statistics, and Stochastic Processes. Hoboken*, NJ, Wiley & Son. 2005.

MARK PERAKH
10106 SAGE HILL WAY,,
ESCONDIDO, CA 92026, USA,
TEL.: 760 751 9932
E-MAIL: <MARPERAK@COX.NET>

Lexical Resources

IJCLA VOL. 3, NO. 2, JUL-DEC 2012, PP. 47–62
RECEIVED 31/10/11 ACCEPTED 09/12/11 FINAL 28/09/12

An Automatic Method for Creating a Sense-Annotated Corpus Harvested from the Web

VERENA HENRICH, ERHARD HINRICHS, AND TATIANA VODOLAZOVA

University of Tübingen, Germany

ABSTRACT

This paper reports on an automatic and language-independent method for compiling a sense-annotated corpus of web data. To validate its language-independence, the method has been applied to English and German. The sense inventories are taken from the Princeton WordNet for English and from the German wordnet GermaNet. The web-harvesting utilizes existing mappings of WordNet and GermaNet to the English and German versions of the web-based dictionary Wiktionary, respectively. The data obtained by this method have resulted in the English WebCAP (short for: Web-Harvested Corpus Annotated with Princeton WordNet Senses) and the German WebCAGe (short for: Web-Harvested Corpus Annotated with GermaNet Senses) resources.

KEYWORDS: *Sense-annotated corpus, sense-tagged corpus, word sense disambiguation, WSD, Princeton WordNet, GermaNet, Wictionary*

1 INTRODUCTION

Sense-annotated corpora are an important resource for a variety of natural language processing tasks including word sense disambiguation, machine translation, and information retrieval. In past resource building, sense-annotated corpora have typically been constructed manually. This has

made the compilation of such resources costly and has put a natural limit on the size of such data sets. This in turn suggests that alternatives to manual annotation need to be explored and automatic, language-independent means of creating sense-annotated corpora need to be investigated. The purpose of this paper is therefore threefold:

1. To propose an automatic method for harvesting and sense-annotating data from the web.
2. To prove the viability and the language-independence of the proposed approach.
3. To make the resulting sense-annotated corpora freely available for other researchers.

The proposed method relies on the following resources as input: (i) a sense inventory and (ii) a mapping between the sense inventory in question and a web-based resource such as Wiktionary[1] or Wikipedia[2].

As a proof of concept and to validate its language-independence, this automatic method has been applied to two languages: To English, a language for which several sense-annotated corpora are already available, as well as to German, a language for which sense-annotated corpora are still in short supply. The sense inventories are taken from the Princeton WordNet for English [1] and from the German wordnet GermaNet [2, 3]. In order to be able to compare the resulting resources for the two languages, the web-harvesting for both languages relies on existing mappings of the wordnets in question with the English and German versions of the web-based dictionary Wiktionary described in [4] and [5], respectively. The resulting resources consist of the web-harvested corpora WebCAP (short for: *Web-Harvested Corpus Annotated with Princeton WordNet Senses*) and WebCAGe (short for: *Web-Harvested Corpus Annotated with GermaNet Senses*). These resources will be made freely available.[3]

The remainder of this paper is structured as follows: An overview of related work is given in Section 2. Section 3 introduces the three resources WordNet, GermaNet, and Wiktionary used in the present research. The algorithm for automatically harvesting and sense-annotating

[1] `http://www.wiktionary.org/`

[2] `http://www.wikipedia.org/`

[3] See `http://www.sfs.uni-tuebingen.de/en/general-and-computational-linguistics/resources/corpora/webcap` and `http://www.sfs.uni-tuebingen.de/en/general-and-computational-linguistics/resources/corpora/webcage`

textual materials from the web is described in Section 4. Section 5 evaluates the proposed approach applied to English and German, and compares the results for the two languages. Finally, the paper concludes with a summary of the results and with an outlook to future work in Section 6.

2 Related Work

With relatively few exceptions to be discussed shortly, the construction of sense-annotated corpora has focussed on purely manual methods. This is true for SemCor, the WordNet Gloss Corpus, and for the training sets constructed for English as part of the SensEval and SemEval shared task competitions [6–8]. Purely manual methods were also used for the German sense-annotated corpora constructed by Broscheit et al. [9] and Raileanu et al. [10] as well as for other languages including the Bulgarian and the Chinese sense-tagged corpora [11, 12]. The only previous attempts of harvesting corpus data for the purposes of constructing a sense-annotated corpus is the semi-supervised method developed by Yarowsky [13], the knowledge-based approach of Leacock et al. [14], later also used by Agirre and Lopez de Lacalle [15], and the automatic association of Web directories (from the Open Directory Project, ODP) to WordNet senses by Santamaría et al. [16].

The latter study [16] is closest in spirit to the approach presented here. It also relies on an automatic mapping between WordNet senses and a second web resource. While our approach is based on automatic mappings between WordNet/GermaNet and Wiktionary, their mapping algorithm maps WordNet senses to ODP subdirectories. Since these ODP subdirectories contain natural language descriptions of websites relevant to the subdirectory in question, this textual material can be used for harvesting sense-specific examples.

The approach of Yarowsky [13] first collects all example sentences that contain a polysemous word from a very large corpus. In a second step, a small number of examples that are representative for each of the senses of the polysemous target word is selected from the large corpus created in step 1. These representative examples are manually sense-annotated and then fed into a decision-list supervised WSD algorithm as a seed set for iteratively disambiguating the remaining examples collected in step 1. The selection and annotation of the representative examples in Yarowsky's approach is performed completely manually and is therefore limited to the amount of data that can reasonably be annotated by hand.

Leacock et al., Agirre and Lopez de Lacalle, and Mihalcea and Moldovan [14, 15, 17] propose a set of methods for automatic harvesting of web data for the purposes of creating sense-annotated corpora. By focusing on web-based data, their work resembles the research described in the present paper. However, the underlying harvesting methods differ. While our approach relies on a wordnet to Wiktionary mapping, their approaches all rely on the monosemous relative heuristic. Their heuristic works as follows: In order to harvest corpus examples for a polysemous word, the WordNet relations such as synonymy and hypernymy are inspected for the presence of unambiguous words, i.e., words that only appear in exactly one synset. The examples found for these monosemous relatives can then be sense-annotated with the particular sense of its ambiguous word relative. In order to increase coverage of the monosemous relatives approach, Mihalcea and Moldovan [17] have developed a gloss-based extension, which relies on word overlap of the gloss and the WordNet sense in question for all those cases where a monosemous relative is not contained in the WordNet dataset.

The approaches of Leacock et al., Agirre and Lopez de Lacalle, and Mihalcea and Moldovan as well as Yarowsky's approach provide interesting directions for further enhancing the WebCAP and WebCAGe resources (for some preliminary discussion on such an integration see Section 6 below).

In our own previous research, we have addressed the issue of automatically creating sense-annotated corpora for German. The creation of the resource WebCAGe described in the present paper relies on a mapping between GermaNet and the German Wiktionary [5] and is based on an earlier study [18]. With WikiCAGe, we have built a second sense-annotated corpus for German [19]. It consists of examples harvested from the German Wikipedia and was constructed by means of an automatic mapping between GermaNet and the German Wikipedia.

3 RESOURCES

3.1 *WordNet and GermaNet*

Both the Princeton WordNet for English [1] and the German wordnet GermaNet [2, 3] are lexical semantic networks that partition the lexical space into sets of concepts that are interlinked by semantic relations such as hypernymy, part-whole relations, entailment, causation, or antonymy. Wordnets are hierarchically structured in terms of the hypernymy relation. A semantic concept is modeled by a *synset*. A synset is a set of

words (called *lexical units*) where all the words are taken to have (almost) the same meaning. Thus a synset is a set-representation of the semantic relation of synonymy, which means that it consists of a list of lexical units.

The Princeton WordNet has served as inspiration and as best practice example for the construction of GermaNet as well as for the creation of other wordnets for a large number of typology diverse languages.[4]

The coverage of the Princeton WordNet includes the four word classes of adjectives, adverbs, nouns, and verbs. Its release 3.0 covers 206,941 word senses, which are grouped into 117,659 synsets. GermaNet covers the three word classes of adjectives, nouns, and verbs. GermaNet's version 6.0 (release of April 2011) covers 93,407 lexical units, which are grouped into 69,594 synsets.

3.2 *Wiktionary*

Wiktionary is a web-based dictionary that is available for many languages, including English and German. As is the case for its sister project Wikipedia, Wiktionary is constructed by contributions of a large number of volunteers and is freely available. The dictionary provides information such as part-of-speech, hyphenation, possible translations, inflection, etc. for each word. It covers, among others, the word categories of adjectives, adverbs, nouns, and verbs. Distinct word senses are distinguished by sense descriptions, accompanied with example sentences illustrating the usage of the sense in question. Further, Wiktionary provides relations to other words, e.g., in the form of synonyms, antonyms, hypernyms, hyponyms, holonyms, and meronyms. Different from WordNet and GermaNet, the relations are (mostly) not disambiguated.

Since Wiktionary is a dynamic resource, it is important to clearly identify the versions used for the present research. The construction of WebCAP is based on a dump of the English Wiktionary as of April 3, 2010, which consists of 335,748 English words comprising 421,847 word senses [4]. For WebCAGe, the German Wiktionary as of February 2, 2011 is utilized, consisting of 46,457 German words and 70,339 word senses [5]. The Wiktionary data is extracted by the freely available Java-based library JWKTL[5].

[4] See `http://www.globalwordnet.org/` for an informative overview.
[5] `http://www.ukp.tu-darmstadt.de/software/jwktl`

4 Creating a Sense-Annotated Corpus Harvested from the Web

The starting point for creating the English WebCAP (short for: *Web-Harvested Corpus Annotated with Princeton WordNet Senses*) and the German WebCAGe (short for: *Web-Harvested Corpus Annotated with GermaNet Senses*) resources are existing mappings of senses in WordNet and GermaNet with Wiktionary senses as described in [4] and [5], respectively. These mappings were created by automatic word sense alignment algorithms with high accuracy: 91.5% for English [4] and 93.8% for German [5]. For German, a manual post-correction step of the automatic alignment was performed that further improved the accuracy of the mapping.

4.1 *Web-Harvesting Sense-Annotated Materials*

Fig. 1 illustrates the existing WordNet-Wiktionary mapping using the example word *crutch*. The polysemous word *crutch* has two distinct senses in WordNet which directly correspond to two separate senses in the English Wiktionary[6]. Each Wiktionary sense entry contains a definition and one or more example sentences illustrating the sense in question. Since the target word in the example sentences for a particular Wiktionary sense (rendered in Fig. 1 in bold face) is linked to a WordNet sense via the sense mapping of WordNet to Wiktionary, the example sentences are automatically sense-annotated and can be included as part of WebCAP.

An example for the GermaNet-Wiktionary mapping using the example word *Option* is given in Fig. 2. As is the case for the English example *crutch*, the polysemous word *Option* has two distinct senses in GermaNet which directly correspond to two separate senses in the German Wiktionary. Again, each Wiktionary sense contains one or more example sentences, which can directly be mapped to a specific sense in GermaNet and thus be sense-annotated and included in WebCAGe. Furthermore, the examples in turn are linked to external references, including sentences contained in Wikipedia articles (see link in the second Wiktionary sense entry in Fig. 2) and in other web-based textual sources such as online newspaper materials and the German Gutenberg text archive[7] (see the topmost sense entry in Fig. 2).

[6] Note that there is one further sense in Wiktionary not displayed here for reasons of space.

[7] `http://gutenberg.spiegel.de/`

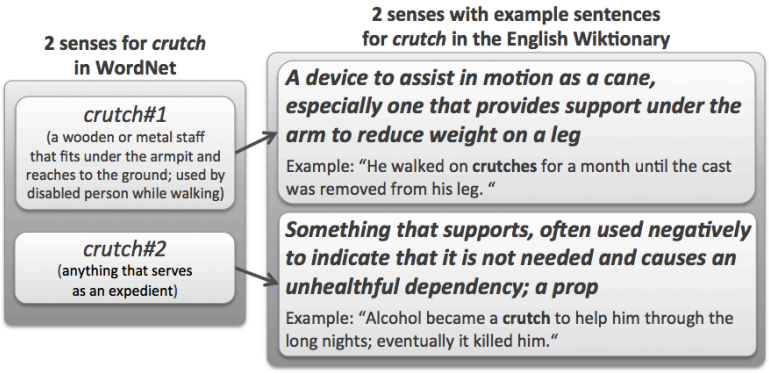


Fig. 1. Sense mapping of WordNet and Wiktionary using the example of *crutch*.

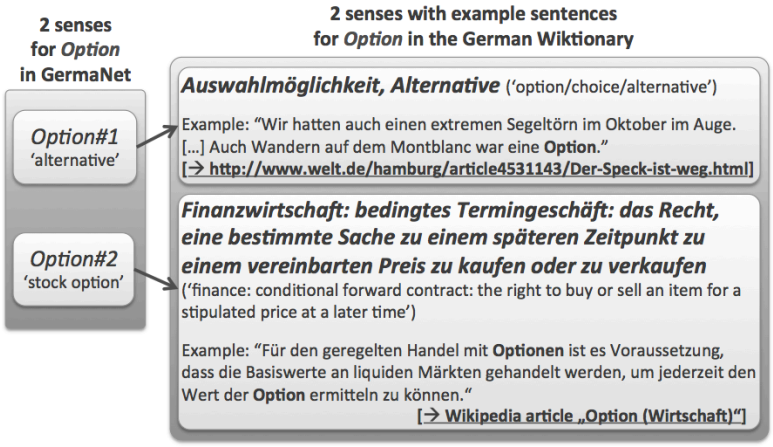


Fig. 2. Sense mapping of GermaNet and Wiktionary using the example of *Option*.

Additional data for WebCAGe is harvested by following the links to Wikipedia and other web-based resources referenced by Wiktionary. Since these links belong to particular Wiktionary sense entries that in turn are mapped to GermaNet senses, the target words contained in these materials are automatically sense-annotated.

Notice that the target word often occurs more than once in a given text. In keeping with the widely used heuristic of "one sense per discourse", multiple occurrences of a target word in a given text are all as-

signed to the same wordnet sense. An inspection of the annotated data shows that this heuristic proves to be highly reliable in practice.[8]

WebCAP and WebCAGe are developed primarily for the purpose of the word sense disambiguation task. Therefore, only those target words that are ambiguous are included in these resources. For the German WebCAGe, this means that each target word has at least two GermaNet senses, i.e., belongs to at least two distinct synsets in GermaNet. For the English WebCAP, each target word has at least two senses in WordNet regardless of word class; i.e., the target word belongs to at least two distinct synsets in WordNet which may belong to more than one word class. Taking into account polysemy across word classes is important for English. In contrast to German, this type of conversion involving the same orthography for different word classes with possibly distinct meanings is a frequent phenomenon in English.

Both the WordNet-Wiktionary and the GermaNet-Wiktionary mappings are not always one-to-one. For example, sometimes one WordNet/GermaNet sense is mapped to more than one sense in Wiktionary. In those cases, all example sentences from all mapped Wiktionary senses are assigned to the WordNet/GermaNet sense in question.

4.2 *Target Word Identification*

The next step for creating a sense-annotated corpus is the target word identification. For highly inflected languages such as German, target word identification is more complex compared to languages with a simplified inflectional morphology such as English and requires automatic lemmatization. Moreover, the target word in a text to be sense-annotated is not always a simplex word, but can also appear as subpart of a complex word such as a compound. Since the constituent parts of a compound are not separated by blank spaces or hyphens, German compounding poses a particular challenge for target word identification. Another challenging case for automatic target word detection in German concerns particle verbs such as *an-kündigen* 'announce'. Here, the difficulty arises when the verbal stem (e.g., *kündigen*) is separated from its particle (e.g., *an*) in German verb-initial and verb-second clause types.

[8] Henrich et al. [18] show that for German the heuristic works correctly in 99.96% of all target word occurrences in the Wiktionary example sentences, in 96.75% of all occurrences in the external webpages, and in 95.62% of the Wikipedia files.

```
Radioaktivität, radioaktiver <tag luids="l88831" lemma="Zerfall"
1 wcat="NN">Zerfall</tag> oder Kern<tag luids="l88831"
2 lemma="Zerfall" wcat="NN">zerfall</tag> ist die Eigenschaft
  instabiler Atomkerne, sich spontan unter Energieabgabe
  umzuwandeln. [...]

  Der Zeitpunkt eines radioaktiven <tag luids="l88831"
3 lemma="Zerfall" wcat="NN">Zerfalls</tag> ist im Voraus nicht
  bestimmbar. [...]

  Im Allgemeinen sind die <tag luids="l88831" lemma="Zerfall"
4 wcat="NN">Zerfall</tag>sprodukte nicht stabil. In den meisten
  Fällen sind die Tochterkerne ihrerseits wieder radioaktiv und
  zerfallen gemäß ihrer eigenen Halbwertszeiten. Auf diese Weise
  entsteht eine Abfolge von radioaktiven <tag luids="l88831"
5 lemma="Zerfall" wcat="NN">Zerfällen</tag>, bis schließlich ein
  stabiler Kern als Endprodukt übrig bleibt. Diese Aufeinander-
  folge radioaktiver <tag luids="l88831" lemma="Zerfall"
6 wcat="NN">Zerfälle</tag> heißt <tag luids="l88831"
7 lemma="Zerfall" wcat="NN">Zerfall</tag>sreihe.[...]

            Source: http://de.wikipedia.org/wiki/Radioaktivität
```

Fig. 3. Excerpt from Wikipedia article *Radioaktivität* 'radioactivity' tagged with the target word *Zerfall* 'radioactive decay'.

As a preprocessing step for target word identification, the web-harvested texts are split into individual sentences, tokenized, and lemmatized. For this purpose, the sentence detector and the tokenizer of the suite of Apache OpenNLP tools[9] and the TreeTagger [20] are used both for English and German. Further, for German, compounds are split by using BananaSplit[10]. Since the automatic lemmatization obtained by the tagger (and the compound splitter) are not a 100% accurate, target word identification also utilizes the full set of inflected forms for a target word whenever such information is available in Wiktionary.

Fig. 3 shows a German example of a sense-annotated text for the target word *Zerfall* in the sense of 'radioactive decay'. The text is an excerpt from the Wikipedia article *Radioaktivität* 'radioactivity' and contains many occurrences of the target word (rendered in bold face). Only the first occurrence shown in Fig. 3 (marked with a `1` on the left margin) exactly matches the word *Zerfall* as is. All other occurrences are either the genitive form *Zerfalls* (occurrence `3`), the genitive plural *Zerfälle* (occurrence `6`), the dative plural *Zerfällen* (occurrence `5`), or part of a compound such as *Kernzerfall*, *Zerfallsprodukte*, or *Zerfallsreihe* (occurrences `2`, `4`, and `7`).

[9] `http://incubator.apache.org/opennlp/`
[10] `http://niels.drni.de/s9y/pages/bananasplit.html`

4.3 *Data Encoding*

For expository purposes, the data format shown in Fig. 3 has been simplified compared to the actual XML data encoding used for both WebCAP and WebCAGe. This data encoding is inspired by the best practise format for sense-annotated corpora established by the sense-annotated corpora used in the SensEval and SemEval shared task competitions [6–8].

Fig. 3 illustrates the information provided for each sense-annotated target word in WebCAGe: (i) a sense ID referring to a lexical unit in GermaNet, (ii) the lemma of the target word, and (iii) the word class of the target word. The target word information in WebCAP following exactly the same data format. However, in the case of WebCAP, the sense information of each target word points to a WordNet synset rather than a WordNet lexical unit. The reason for this difference in encoding stems from the WordNet/GermaNet-Wiktionary mappings: The WordNet-Wiktionary mapping links synset IDs in WordNet to Wiktionary senses, whereas the GermaNet-Wiktionary mapping links lexical unit IDs in GermaNet to Wiktionary senses.

5 Evaluation and Discussion of the Results

In order to assess the effectiveness of the approach, we examine and compare the overall sizes of WebCAP and WebCAGe (see Table 1) and present a precision and recall based evaluation for the algorithm that is used for automatically identifying the target words in the harvested texts (see Table 2).

The target words in WebCAP belong to 3628 distinct polysemous words contained in WordNet, among which there are 934 adjectives, 174 adverbs, 1480 nouns, and 1040 verbs. These words have on average 3.7 senses in WordNet (1.9 for adjectives, 2.6 for adverbs, 4.1 for nouns, and 5.0 for verbs). The target words in WebCAGe belong to 2607 distinct polysemous words contained in GermaNet (211 adjectives, 1499 nouns, and 897 verbs) which have on average 2.9 senses in GermaNet (2.4 for adjectives, 2.6 for nouns, and 3.6 for verbs).

Table 1 shows the overall sizes of WebCAP and WebCAGe: The numbers of tagged word tokens (i.e., the target word occurrences), the number of sentences containing those tags, and the number of overall sentences (i.e., all sentences in the corpora including those where no target word has been tagged) separately for the four word classes of adjectives, adverbs, nouns, and verbs. The numbers for WebCAP describe the Wiktionary

example sentences only, whereas the numbers for WebCAGe are given separately for the Wiktionary example sentences (in order to be comparable with WebCAP), for the external materials, and overall (the sum of the Wiktionary example sentences and the external materials). WebCAGe contains a total of 10750 tagged word tokens whereas WebCAP only contains 6526 word tokens. Even if we compare the numbers of the Wiktionary example sentences in WebCAP (6526 tagged word tokens) with those in WebCAGe (7644 tagged word tokens), i.e., excluding the external materials from WebCAGe, the German resource is larger than the English one. This is especially astonishing considering that the English input resources constitute a multiple of their German counterparts: The Princeton WordNet contains 1.7 times as many word senses as GermaNet and the English Wiktionary contains 6 times as many word senses as the German Wiktionary (see Section 3). The explanation for the German Wiktionary examples outnumbering those for English has to do with the online instructions given to Wiktionary contributors for English. For the English Wiktionary, contributors are asked to accompany each word sense definition by a quotation that illustrates the definition in question and to compose example sentences on their own only if no suitable quotation sentence can be found.[11] Accordingly, the English Wiktionary contains fewer example sentences compared to German.

According to the guidelines for the English Wiktionary, a *quotation* is an attested example taken from a literary work or from some other published textual material. Such quotations are accompanied by the appropriate reference to their textual source. The version of the API that was used to extract the Wiktionary data does not support the harvesting of the quotations themselves and the textual sources from which those quotations are taken. We anticipate that the size of WebCAP would increase significantly if the harvesting functionality is extended to the set of quotations that contributors are encouraged to provide for each sense definition. For the German Wiktionary, the situation is different in that example sentences are a mixture of made-up materials and attested examples that are often cross-referenced with their online sources and can thus be harvested automatically by the API.

It is also noticeable that the relative numbers of the different word classes are rather equally distributed in WebCAP, whereas there are con-

[11] See `http://en.wiktionary.org/wiki/Wiktionary:Entry_layout_explained` for the relevant instructions.

Table 1. Current sizes of WebCAP and WebCAGe.

		WebCAP	WebCAGe		
		Wiktionary examples	**Wiktionary examples**	**External materials**	**All texts**
Number of tagged word tokens	adjectives	1522	575	138	713
	adverbs	311	0	0	0
	nouns	2596	4103	2744	6847
	verbs	2097	2966	224	3190
	all word classes	6526	7644	3106	10750
Number of tagged sentences	adjectives	1488	565	133	698
	adverbs	302	0	0	0
	nouns	2526	3965	2448	6413
	verbs	2056	2945	224	3169
	all word classes	6372	7475	2805	10280
Total number of sentences	adjectives	1578	623	66757	67380
	adverbs	317	0	0	0
	nouns	2726	4184	392640	396824
	verbs	2181	3087	152303	155390
	all word classes	6802	7894	611700	619594

siderably more texts in WebCAGe contributed by nouns than by adjectives and verbs (see Table 1).[12]

Apart from the size of the resources in question, the usefulness of the compiled data sets depends crucially on the quality of the annotated data. WebCAP and WebCAGe are the results of an automatic harvesting method. Such an automatic method will only constitute a viable alternative to the labor-intensive manual method of creating sense-annotated corpora if the results are of sufficient quality so that the harvested data set can be used as is or can be further improved with a minimal amount of manual post-editing. For the purposes of the present evaluation, a precision and recall based analysis was conducted, and the tagged target words are manually verified. For WebCAGe, all textual materials have been manually checked, while for WebCAP, only the first 1,000 Wiktionary example sentences for nouns and the first 500 sentences for adjectives, adverbs, and verbs could be manually verified. Table 2 shows that precision and recall for all word classes are above 97% in WebCAP and above 93% in WebCAGe. The only deviations are the results for verbs

[12] The reason why there are no tagged adverbs in WebCAGe is due to the GermaNet resource which covers adjectives, nouns, and verbs, but no adverbs.

Table 2. Evaluation of the algorithm of identifying the target words.

			WebCAGe		
		WebCAP	**Wiktionary examples**	**External materials**	**All texts**
Precision	adjectives	97.98%	97.70%	98.39%	98.21%
	adverbs	98.68%	–	–	–
	nouns	97.62%	98.17%	95.52%	96.18%
	verbs	97.88%	97.38%	87.37%	89.80%
	all word classes	97.90%	97.32%	93.29%	94.30%
Recall	adjectives	99.19%	97.70%	97.48%	97.54%
	adverbs	99.01%	–	–	–
	nouns	99.27%	98.30%	95.37%	96.10%
	verbs	98.99%	97.51%	96.26%	96.58%
	all word classes	99.16%	97.94%	96.36%	96.01%

that occur in WebCAGe, which are slightly lower than the results for the other word classes. Apart from this one exception, the results in Table 2 prove the viability of the proposed method for automatic harvesting of sense-annotated data. The average precision for all three word classes is of sufficient quality to be used as is if approximately 2-5% noise in the annotated data is acceptable. In order to eliminate such noise, manual post-editing would be required.

6 Conclusion and Future Work

This paper has described an automatic method for harvesting and sense-annotating data from the web. In order to validate the language-independence of the approach, the proposed method has been applied to both English and German. The publication of this paper will be accompanied by making the two sense-annotated corpora WebCAP and WebCAGe freely available. In the case of WebCAGe, the automatic sense-annotation of all target word has been manually verified.

In order to further enlarge the WebCAP and WebCAGe resources, it would be interesting and worthwhile to use the automatically harvested sense-annotated examples as the seed set for Yarowsky's iterative method for creating a large sense-annotated corpus. Another fruitful direction for further automatic expansion of WebCAP and WebCAGe consists of using the heuristic of monosemous relatives used by Leacock et al., by Agirre

and Lopez de Lacalle, and by Mihalcea and Moldovan. However, we have to leave both of these matters for future research.[13]

Finally, we plan to apply our method to further languages. A precondition for such an experiment are existing mappings between the sense inventories in question and web-based resources such as Wiktionary or Wikipedia. With BabelNet, Navigli and Ponzetto [21] have created a multilingual resource that allows the testing of our approach with languages other than English and German.

ACKNOWLEDGEMENTS We are very grateful to Emanuel Dima, Yana Panchenko, Klaus Suttner, and Yannick Versley for their support in obtaining the external web-based materials. We would like to thank Reinhild Barkey, Sarah Schulz, and Johannes Wahle for their help with the evaluation. Special thanks go to Christian M. Meyer, who has provided both the English Wiktionary and the JWKTL API in the same versions that were used for the WordNet-Wiktionary mapping, and to Tristan Miller, who provided helpful input to the final data format of WebCAGe. This work was supported by the CLARIN-D grant of the BMBF and the SFB 833 grant of the DFG.

REFERENCES

1. Fellbaum, C., ed.: WordNet – An Electronic Lexical Database. The MIT Press (1998)
2. Henrich, V., Hinrichs, E.: GernEdiT – the GermaNet editing tool. In: Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta. (2010) 2228–2235
3. Kunze, C., Lemnitzer, L.: GermaNet – representation, visualization, application. In: Proceedings of the 3rd International Language Resources and Evaluation (LREC'02), Las Palmas, Canary Islands. (2002) 1485–1491
4. Meyer, C.M., Gurevych, I.: What psycholinguists know about chemistry: Aligning Wiktionary and WordNet for increased domain coverage. In: Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP), Chiang Mai, Thailand. (2011) 883–892
5. Henrich, V., Hinrichs, E., Vodolazova, T.: Semi-automatic extension of GermaNet with sense definitions from Wiktionary. In: Proceedings of the 5th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC'11), Poznan, Poland. (2011) 126–130

[13] For a description of these approaches, see Section 2.

6. Agirre, E., Marquez, L., Wicentowski, R.: Proceedings of the 4th International Workshop on Semantic Evaluations. Assoc. for Computational Linguistics, Stroudsburg, PA, USA (2007)
7. Erk, K., Strapparava, C.: Proceedings of the 5th International Workshop on Semantic Evaluation. Assoc. for Computational Linguistics, Stroudsburg, PA, USA (2010)
8. Mihalcea, R., Chklovski, T., Kilgarriff, A.: Proceedings of Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona, Spain. Association for Computational Linguistics (2004)
9. Broscheit, S., Frank, A., Jehle, D., Ponzetto, S.P., Rehl, D., Summa, A., Suttner, K., Vola, S.: Rapid bootstrapping of word sense disambiguation resources for German. In: Proceedings of the 10th Konferenz zur Verarbeitung Natürlicher Sprache, Saarbrücken, Germany. (2010) 19–27
10. Raileanu, D., Buitelaar, P., Vintar, S., Bay, J.: Evaluation corpora for sense disambiguation in the medical domain. In: Proceedings of the 3rd International Language Resources and Evaluation (LREC'02), Las Palmas, Canary Islands. (2002) 609–612
11. Koeva, S., Leseva, S., Todorova, M.: Bulgarian sense tagged corpus. In: Proceedings of the 5th SALTMIL Workshop on Minority Languages: Strategies for Developing Machine Translation for Minority Languages, Genoa, Italy. (2006) 79–87
12. Wu, Y., Jin, P., Zhang, Y., Yu, S.: A Chinese corpus with word sense annotation. In: Proceedings of 21st International Conference on Computer Processing of Oriental Languages (ICCPOL'06), Singapore. (2006) 414–421
13. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics (ACL'95), Stroudsburg, PA, USA, Association for Computational Linguistics (1995) 189–196
14. Leacock, C., Chodorow, M., Miller, G.A.: Using corpus statistics and wordnet relations for sense identification. Computational Linguistics **24**(1) (1998) 147–165
15. Agirre, E., Lopez de Lacalle, O.: Publicly available topic signatures for all WordNet nominal senses. In: Proceedings of the 4th International Conference on Languages Resources and Evaluations (LREC'04), Lisbon, Portugal. (2004) 1123–1126
16. Santamaría, C., Gonzalo, J., Verdejo, F.: Automatic association of web directories to word senses. Computational Linguistics **29**(3) (2003)
17. Mihalcea, R., Moldovan, D.: An automatic method for generating sense tagged corpora. In: Proceedings of the American Association for Artificial Intelligence (AAAI'99), Orlando, Florida. (1999) 461–466
18. Henrich, V., Hinrichs, E., Vodolazova, T.: WebCAGe – a web-harvested corpus annotated with GermaNet senses. In: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL'2012), Avignon, France. (2012) 387–396

19. Henrich, V., Hinrichs, E., Suttner, K.: Automatically linking GermaNet to Wikipedia for harvesting corpus examples for GermaNet senses. Journal for Language Technology and Computational Linguistics (JLCL) **27**(1) (2012) 1–19
20. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Proceedings of the International Conference on New Methods in Language Processing, Manchester, UK. (1994)
21. Navigli, R., Ponzetto, S.P.: BabelNet: Building a very large multilingual semantic network. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10), Uppsala, Sweden. (2010) 216–225

VERENA HENRICH
DEPARTMENT OF LINGUISTICS,
UNIVERSITY OF TÜBINGEN,
WILHELMSTR. 19, 72074 TÜBINGEN, GERMANY
E-MAIL: <VERENA.HENRICH@UNI-TUEBINGEN.DE>

ERHARD HINRICHS
DEPARTMENT OF LINGUISTICS,
UNIVERSITY OF TÜBINGEN,
WILHELMSTR. 19, 72074 TÜBINGEN, GERMANY
E-MAIL: <ERHARD.HINRICHS@UNI-TUEBINGEN.DE>

TATIANA VODOLAZOVA
DEPARTMENT OF LINGUISTICS,
UNIVERSITY OF TÜBINGEN,
WILHELMSTR. 19, 72074 TÜBINGEN, GERMANY
E-MAIL: <TATIANA.VODOLAZOVA@UNI-TUEBINGEN.DE>

IJCLA VOL. 3, NO. 2, JUL-DEC 2012, PP. 63–76
RECEIVED 30/10/11 ACCEPTED 09/12/11 FINAL 08/06/12

Mapping Synsets in WordNet to Chinese

SHI WANG

Chinese Academy of Sciences, China

ABSTRACT

WordNet is a large lexical database which has important influence on many computational linguistics related applications, but unfortunately cannot be used in other languages except English. This paper presents an automatic method to map WordNet synsets to Chinese, and then generate an homogeneous Chinese WordNet. The proposed approach is grounded on the viewpoint that most cognitive concepts are languages independent, and can be mapped from one language to another unambiguously. Firstly, we utilize offline/online English-Chinese lexicons and term translation system to translate the words in WordNet. One English word is translated to multiple Chinese words, and one synsets is translated to a group of Chinese words. We secondly cluster these Chinese words into synonym-sets according to their senses. And finally, we select the right synonym-set for given synset. We regard the proper word-set choosing process as a classifier problem, and put forward 9 classifying features based on relations in WordNet, Chinese morphologies, and translation intersections. Besides, an lexico-syntactic patterns based heuristic rule is combined for higher recall. Experiment results on WordNet 3.0 show the overall synsets translating coverage of out method is 85.12% with the precision of 81.37%.

KEYWORDS: *WordNet translation, Chinese WordNet, lexical resources, computational linguistics*

1 INTRODUCTION

WordNet is a widely-used large-scale lexical database in which nouns, verbs, adjectives and adverbs are grouped into sets of cognitive concepts

(also called synsets) [1]. Synsets are interlinked by means of conceptual semantic relationships and then construct a net. Up to now, there are totally 155,287 words and 117,659 synsets in WordNet 3.0.

WordNet has been used in a large range of applications including natural language process, information retrieval, word sense disambiguation, text classification, image retrieval, etc. Unfortunately, this valuable resource cannot be directly used in other languages except English.

This paper introduces an automatic method for the construction of Chinese WordNet by mapping WordNet synsets to Chinese. The root of our work is that most synsets are languages independent and can be directly mapped to other languages unambiguously, though words in synsets may not be explicitly one by one translated. Most of synsets in WordNet, which express cognitive concepts in real world, can also be expressed by Chinese. If we map all synsets to Chinese, we obtain Chinese WordNet in which synsets are interlinked by identical semantic relations as in WordNet.

We firstly utilize offline/online English-Chinese lexicons and term translation system to translate the words in WordNet. One English word is translated to multiple Chinese words, and one synsets is translated to a group of Chinese words. We secondly cluster these Chinese words into synonym-sets according to their senses. And finally, we select the right synonym-set for given synset.

Regarding the proper word-set choosing process as a classifier problem, we put forward 9 classifying features based on relations in WordNet, Chinese morphologies, and translation intersections. Besides, an lexico-syntactic patterns based heuristic rule is combined for higher recall. Experiment results on WordNet 3.0 show the overall synsets translating coverage of out method is 85.12% with the precision of 81.37%. Experiment data and final results is available from `http://www.knowology.cn/cicling12/ChWordNet.rar` and `http://www.cicling.org/2012/data/33`.

The remainder of the paper is organized as follows. In section 2 we present related work. Section 3 described the proposed method in detail and section 4 gives its experimental results. Finally, we discuss shortcomings of our work and conclude this paper.

2 Related Work

The Global WordNet Association Association [2] provide a free, public and non-commercial organization that provides a platform for discussing,

sharing and connecting WordNets for all languages in the world. The Association held a conference every two years.

EruoWordNet has been built according to same structure with WordNet[3]. EuroWordNet is a multilingual database with WordNet for several European languages including Dutch, Italian, Spanish, German, French, Czech and Estonian, and are structured in the same way as the WordNet.

In Asia [4] shows an evaluation of the Korean WordNet. The purpose of their work is to study how well the manually created lexical taxonomy is built. Evaluation is done level by level, and the reason for selecting words for each level is that we want to compare each level and to find relations between them.

For Chinese, CiLin [5] and HowNet [6] are analogous but very different resources. CiLin has a four-layer semantic structure but does not provide clear relations between words. HowNet is an extra-linguistic knowledge base which unveils inter-concept relations and inter-attribute relations of the concepts. It uses sememes to explain all the concepts and relations in it, which is different from the relational analysis methodology adopted by WordNet. [7] and [8] integrated CiLin and HowNet with WordNet.

Because built manually requires great efforts, much work focused on automatical WordNet translation these years. [9] proposes a method to map Chinese words into WordNet by integrating five linguistic resources including English/Chinese sense-tagged corpora, English/Chinese thesauruses, and a bilingual dictionary. A Chinese WordNet and a Chinese-English WordNet are derived from the structures of WordNet.

[10] uses a statistics-based method that looks for the intersection of word sense to translate of synset of WordNet. [11] describes automatic techniques for mapping entries to WordNet senses.

[12] examines the validity of cross-lingual lexical semantic relations inferences by bootstrapping a Chinese WordNet. They claim that such correspondences must be based on lexical semantic relations, rather than top ontology or word translations.

3 METHOD

In brief, we firstly translate synsets into a group of Chinese synonym-sets based on word translations, and then select right one for given synset. Taking synset “tiger, Panthera tigris – (large feline of forests in most of Asia having a tawny coat with black stripes; endangered)” for instance, there are four steps for mapping it to Chinese:

1. Translating each word in synset to Chinese
   - tiger → 虎/tiger, 公虎/male tiger, 暴徒/mob, 凶徒/villain
   - Panthera tigris → 老虎/tiger, 虎/tiger
2. Clustering translations into synonym-sets according to their senses
   - tiger → {虎/tiger, 公虎/male tiger}, {暴徒/mob, 凶徒/villain}
   - Panthera tigris → {老虎/tiger, 虎/tiger}
3. Choosing right synonym-sets for synset
   - tiger → {虎/tiger, 公虎/male tiger}✓, {暴徒/mob, 凶徒/villain}×
   - Panthera tigris → {老虎/tiger, 虎/tiger}✓
4. Merging the right synonym-sets for synset as result
   - result = {虎/tiger, 公虎/male tiger, 老虎/tiger},

In step 3, symbols ✓/× represent whether the word-set was chosen or not. As a result, synset {tiger, Panthera tigris} is mapped to {虎/tiger, 公虎/male tiger, 老虎/tiger}. We note that semantic relationships which linked with it in WordNet are still unchanged. So if we can map all synsets to Chinese, we obtain an Chinese WordNet.

3.1 *Definitions*

Definition 1. *For a particular sense* ***ss*** *of an English word, its* ***sense translation*** $T_{ss}(ss) = \{cw_1, ..., cw_n\}$ *is a set of Chinese synonyms which express its meaning.*

Example. {虎/tiger, 公虎/male tiger} is a one sense translation for "tiger".

Definition 2. *For an English word* ***ew*** *with* $\mathtt{m}$ *senses* $\{ss_1, ..., ss_m\}$*, its* ***clustered word translations*** $T_{wd}(ew) = \{T_{ss}(ss_1), ..., T_{ss}(ss_m)\}$ *is the set of its senses translations.*

Example. T_{wd}(tiger) $= \left\{ \begin{matrix} \{\text{老虎/tiger, 虎/tiger}\} \\ \{\text{暴徒/mob, 凶徒/villain}\} \end{matrix} \right\}$

Definition 3. *Given an English synset* ***esy*** *of* $\mathtt{m}$ *words* $\{ew_1, ..., ew_m\}$*, its candidate translations* $CT_{esy}(esy) = T_{wd}(ew_1) \cup, ..., \cup T_{wd}(ew_m)$*, are called* ***synset candidate translations****. This the union of its words' translations.*

Example. $CT_{sy}\left(\left\{ \begin{matrix} \text{tiger,} \\ \text{Panthera} \\ \text{tigris} \end{matrix} \right\}\right) = \left\{ \begin{matrix} \{\text{老虎/tiger, 虎/tiger}\} \\ \{\text{暴徒/mob, 凶徒/villain}\} \\ \{\text{老虎/tiger, 虎/tiger}\} \end{matrix} \right\}$

3.2 *Getting Synsets Candidate Translations*

Words translating is the base of our whole approach. Besides common words, there are also lots of multi-word expressions in WordNet, including technical terms (“hydroflumethiazide”), fixed expressions (“by and large”), compound phrases (“car park”), verb-particle constructions (“look up”), and light verbs (“make a face”), etc., which are all difficult to translate using traditional dictionaries.

In order to translate as many words as possible, we utilize 8 resources which are complementary with each other as listed in Table 1.

Table 1. Word translating resources

ID	Resource	Translations clustered according to senses?
1	American Heritage	yes
2	Modern E-C	yes
3	Modern Comprehensive E-C	yes
4	Concise E-C	no
5	Landau E-C	common words: no; terms: yes
6	HaiCi Online[1]	no
7	Google Online[2]	yes
8	TermTrans [13]	yes

When translating words using these resources, we want to cluster translations into synonym-sets which will be used to form Chinese synsets as last. Table 1 also shows whether the resources’ translations have already been clustered or not. Accordingly, we devise words translating procedure.

- **Translating common words.** Given an English word, translating it using dictionaries which have already clustered their translations according to word senses, that is, resources 1, 2, 3, and 7. Clustering translations into word-sets as these dictionaries provide.
- **Translating rarely used words offline.** If not translated, translating using Concise E-C dictionary. Concise E-C has the largest size among all lexicons, and most rarely used words which are not disposed in step 1, such as “harpsichordist”, appear in it. According to Zipf law [14], these rarely used words often have unique sense. So

[1] http://dict.cn

[2] http://translate.google.cn

although translations of Concise E-C are not organized well, the one word translation for rarely used words are adoptable.

- **Translating multi-word expressions offline.** If not translated, translating only using the term translations of Landau E-C dictionary.
- **Translating rarely used words online.** If not translated, translating using HaiCi online dictionary which will automatically transform morphology of word. HaiCi can automatically transform morphologies of words and return related translations. For example, if we look up "antlered" which is not embodied in HaiCi, it will return the translation of "antler" and illuminate that "antlered" is the adjective morphology of "antler" meanwhile. This feature can highly improve the word translation coverage.
- **Translating multi-word expressions online.** If not translated, translating using TermTrans. TermTrans can dispose multi-word expressions. And because most multi-word expressions have unique translation, we only accept the best result TermTrans gives.

We ensure translations are separated according to senses by taking the one-word translation for Concise E-C dictionary, HaiCi online dictionary, and discarding translations for common words in Landau E-C dictionary.

Although resources we adopted are carefully selected, it is inevitable that there are still some words cannot be translated. In experiment section, we will give translation coverage rate in detail.

As shown in definition 3, synsets candidate translations is the union of their containing clustered words translations.

3.3 *Selecting Sense Translations for Synsets*

As presented above, each synset is translated to a group of synonym-sets in which some are right for the synset and others are not. In a special case that there is only one candidates synonym-set, there is no other choice besides accepting it. We call such synsets *clear synsets*. In our experiment, 26.06% synsets in WordNet are clear synsets. For the other synsets, we managed to select right sense translations.

We regarded the selecting procedure as a classifying problem. For a candidate synonym-set, we concluded a group of features to judge whether it is the proper one or not. The features are designed based on relations in WordNet, Chinese morphologies, and translation intersections. A binary classifier was trained using the features introduced below.

INNER-INTERSECTION FEATURE Words in a same synset are synonyms, so their proper translations should share common words. Taking synset "tiger, Panthera tigris" for example, the right sense translations for the two words have a common word "虎/tiger". So if two candidate sense translations have intersections, they are both likely to be the right ones.

We give the explicit measuring function for this feature as follows, which quantifies the shared words number of candidate sense translations in a same synset.

$$F_{II}(T_{ss}(ss_i)) = \\ |\{T_{ss}(ss_j) \in CT_{sy}(esy) | T_{ss}(ss_i) \cap T_{ss}(ss_j) \neq \emptyset|$$

OUTER-INTERSECTION FEATURES In WordNet, SIMILAR-TO (SIM for simplicity) is conceptual relationship which reflects two adjective synsets are similar. For example, "{absorbing, engrossing, fascinating, gripping, riveting}" is similar to "{interesting}".

Being similar is close to being synonymous. So, enlightened by the inner-intersection feature, we proposed outer-intersection feature based on the hypothesis that similar synsets would share common translations. To be specified, for a pair of synsets which satisfied SIM relations, if two candidate translation share some words, the two candidates are both likely to be right ones.

For other two relations SEE-ALSO (SEE) and VERB-GROUP (GRP), we can get analogical features. The three outer-intersections features are calculated as follows:

$$F_{\text{SIM}}(T_{ss}(ss_i)) = \\ |\{T_{ss}(ss_j) \in CT_{sy}(\text{SIM}(esy)) | T_{ss}(ss_i) \cap T_{ss}(ss_j) \neq \emptyset|$$

$$F_{\text{SEE}}(T_{ss}(ss_i)) = \\ |\{T_{ss}(ss_j) \in CT_{sy}(\text{SEE}(esy)) | T_{ss}(ss_i) \cap T_{ss}(ss_j) \neq \emptyset|$$

$$F_{\text{GRP}}(T_{ss}(ss_i)) = \\ |\{T_{ss}(ss_j) \in CT_{sy}(\text{GRP}(esy)) | T_{ss}(ss_i) \cap T_{ss}(ss_j) \neq \emptyset|$$

where {SIM|SEE|GRP}(*esy*) are the {SIM|SEE|GRP} linked synsets of *esy* in WordNet.

LEXICAL CONSTRUCTION FEATURES ATTRIBUTE is a relation between noun synsets and adjective synsets which express that the adjective synsets are attributes of noun synsets. For example, {"able"} is an attribute of {"ability"}.

In Chinese, the nouns plus auxiliary "的/of" is likely to be form its attribute adjectives. We use this word formation rule to judge synsets which are linked by ATTRIBUTE relations. Taking {"able"} and {"ability"} for example,

- CT_{esy}({able}) = {
  {能/able,可/able,会/able},
  {有能力的/capable,能干的/capable,有才能的/able} }
- CT_{esy}({ability}) = {
  {能力/ability,能耐/ability,才能/talent,本领/ability} }

We can easily determine that {有能力的/capable,能干的/capable,有才能的/able} is right for synset {able} because in Chinese, a noun added suffix "的/of" often constructs the corresponding attribute adjective. In the same manner, we can also propose four other lexical features based on HYPERNYM, SISTER, PART-OF and ANTONYM relations.

In Chinese, hypernyms are ofter suffixes of hyponyms (for example, "动物/animal" is hypernym and also suffix of "哺乳动物/mammal"), and then sisters are often share common suffixes ("哺乳动物/mammal" and "爬行动物/reptiles" are in sister synsets, and also share same suffix literally). Parts and wholes sometimes contain same prefixes ("屋顶/roof" is a part of "屋子/house", and they have same prefix), and antonyms can be obtained by simply adding special prefix like "反, 非, 不/aiti-, un-, no-" to words.

$$F_{\text{ATTR}}(T_{ss}(ss_i)) = |\{T_{ss}(ss_j) \in CT_{sy}(\text{ATTR}(esy))|f_a(T_{ss}(ss_i), T_{ss}(ss_j))|$$

$$F_{\text{HYP}}(T_{ss}(ss_i)) = |\{T_{ss}(ss_j) \in CT_{sy}(\text{HYP}(esy))|f_h(T_{ss}(ss_i), T_{ss}(ss_j))|$$

$$F_{\text{SIST}}(T_{ss}(ss_i)) = |\{T_{ss}(ss_j) \in CT_{sy}(\text{SIST}(esy))|f_s(T_{ss}(ss_i), T_{ss}(ss_j))|$$

$$F_{\text{PART}}(T_{ss}(ss_i)) = |\{T_{ss}(ss_j) \in CT_{sy}(\text{PART}(esy))|f_p(T_{ss}(ss_i), T_{ss}(ss_j))|$$

$$F_{\text{ANTI}}(T_{ss}(ss_i)) = |\{T_{ss}(ss_j) \in CT_{wd}(\text{ANTI}(ew))|f_t(T_{ss}(ss_i), T_{ss}(ss_j))|$$

where $f_{\{a,h,s,p,t\}}$ are boolean function described above. Detailed calculating formulas are omitted the sake of brevity.

The above 9 features can be calculated efficiently when classifying synsets candidate translations. In our experiments, we firstly use these features to train a classifier. For the candidates which can not classified, we turn around the following more time-consuming lexico-syntactic patterns rule.

LEXICO-SYNTACTIC PATTERNS FEATURES Lexico-syntactic patterns [15] have the ability to express semantic relationships between concepts, such as "X is a kind of Y" or "X such as Y". In WordNet, all the conceptual relations can be expressed by lexico-syntactic patterns. Then for the ambiguous synsets candidate translations, we can testify them by using such patterns.

For instance, for synset "tiger", $T_{ss}(ss_1)$={虎/tiger, 公虎/male tiger} and $T_{ss}(ss_2)${暴徒/mob, 凶徒/villain}, if we can obtain its hypernym synsets $\{bigcat, cat\}$ whose synset candidate translation is {猫/cat,猫科动物/$felid$}, then we can tell ss_1 is the required one by indexing sentences like "虎是一种猫科动物/tiger is a kind of felid" from corpus.

Using web search engines, we can quickly get the number of snippets which contain certain sentences. In our experiments, we use Google and then restrict our patterns to abide by Google query term expressions. Table 2 displays some of typical patterns we conclude, where c_1 stands for the words in the source synsets and c_2 represents the target synsets' words for a certain relation in WordNet. The double quotation marks that bracket the patterns can make Google search them as whole units, and the wildcards '*' can represent any single word.

For an synset, we firstly find its relative synsets. After filling each word in initial synset and related synsets to corresponding patterns according to their relationship, we feed the query string to Google and judge synset translation by return web pages number.

We did not use the hitting page numbers as features to train a classifier because it is very time costing to get all the numbers for all patterns. A

Table 2. Some lexico-syntactic patterns for synset disambiguation

ID	Relations	Patterns	Patterns in English
01	SYNSET HYPERNYM	c_1是一*c_2	c_1 is a * c_2
02		c_2等c_1	c_2 such as c_1
03	INSTANCE HYPERNYM	c_1属于c_2	c_1 belongs to c_2
04		c_2源自c_1	c_2 is derived from c_1
05	MEMBER-OF	c_1是c_2之一	c_1 is member of c_2
06		c_2中的c_1	c_1 in c_2
07	SUBSTANCE-OF	c_1是c_2的成分	c_1 is substance of c_2
08		c_2由c_1构成	c_2 is made of c_1
09	PART-OF	c_1是c_2的一部分	c_1 is a part of c_2
10		c_2由c_1组成	c_2 is composed of c_1
11	ATTRIBUTE	c_1是c_2的	c_1 is c_2
12		c_2的c_1	c_1 of c_2
13	CAUSE	c_1导致c_2	c_1 cause c_2
14		c_2是因为c_1	c_2 is caused by c_1

empirical method is adopted. That is, if the hitting page number exceeds an experiential threshold for a particular pattern, we accept the candidate translation. If we can query Google or some other huge corpus quickly, we can further use the hits number as features to train the classifier.

3.4 *Merging Selected Sense Translations*

Different dictionaries generate different translations for a same word. For example, for word "tiger", Concise E-C dictionary translates its one sense to {虎/tiger, 公虎/male tiger}, while Modern Comprehensive E-C dictionary gives {老虎/tiger, 虎/tiger}.

So, multi sense translations will be accepted in the candidates choosing procedure. We merged these synonym-sets to generate a compact and integrative translation because they are actually represents same meaning. After merging, we get the right translations for synsets.

In word translating procedure, we have ensured each word-set are synonyms. Being synonymous is transitive for words. So if we merge the word-sets which share common words, the new formed word-set is also a synonym-set.

Our merging strategy is very strict. Another common used method is based on edit distance, that is, merging word-set which have short edit-distances. In experiment, such a relax strategy performs bad. Most Chinese words are very short and might be very different in sense even

they are very similar in morphology. For example, "老虎/tiger" and "老师/teacher" have short edit distance 1, but are completely different in meaning.

4 EXPERIMENT

4.1 *Word Translation Results*

Table 3 shows the word translation percentage for all resources listed in Table 1.

Table 3. Word translation coverage of all the resources

ID	Resource	Coverage
1	American Heritage Dictionary	35.52%
2	Modern E-C dictionary	32.40%
3	Modern comprehensive E-C dictionary	25.81%
4	Concise E-C dictionary	19.75%
5	Landau E-C dictionary	20.14%
6	HaiCi online dictionary	9.55%
7	Google online dictionary	38.72%
8	TermTrans Tool	6.10%
	Average	84.33%

From Tabel 3, we can see that although every distinct resource's coverage is low, the total coverage can reach 84.33%. That means our resources are complementary with each other. And excluding TermTrans, all the other dictionaries are manually compiled and with very high precision.

Errors are mainly caused by the mixing of translations with different senses. For example, in Modern E-C dictionary, word "forefront" are translated to be "最前面/the part in the front or nearest the viewer,最前线/the position of greatest importance or advancement", but these two words are distinguished in WordNet. Table 3 also demonstrates that merging sense translations does not generate too much errors.

4.2 *Synset Candidate Translations Classifying Results*

For different kinds of synsets (noun, verb, adjective and adverb ones), they can utilize different features. Inter-intersection features for VERB-ALSO relations are not available for Noun synsets, for example. So when

constructing trainset, in order to make sure that each feature can be used, we randomly select 200 positive and 200 negative samples which have valid feature value for each feature. There are 1,500 positive and 1,500 negative samples at all, making up about 0.18% for all sense translations.

We adopted NativeBayes, J48, and AdaboostM1 to train the classifier. The labels are 1/0 and results are verified with 10 cross-validation. The performance for all kinds of synsets are shown in Table 4.

Table 4. Result of classifier

Synset	Label	NaiveBayes			J48			AdaboostM1		
		p	r	F1	p	r	F1	p	r	F1
Noun	1	0.921	0.729	0.814	0.85	0.904	**0.876**	0.863	0.866	0.8
	0	0.657	0.893	0.757	0.816	0.726	**0.768**	0.768	0.764	0.766
Verb	1	0.854	0.818	**0.836**	0.873	0.758	0.812	0.854	0.78	0.816
	0	0.861	0.889	**0.875**	0.827	0.912	0.867	0.837	0.894	0.865
Adj	1	0.883	0.852	0.867	0.858	0.887	**0.872**	0.878	0.856	0.867
	0	0.811	0.849	**0.829**	0.837	0.798	0.817	0.813	0.84	0.827
Adv	1	0.904	0.853	**0.878**	0.824	0.891	0.856	0.892	0.853	0.872
	0	0.801	0.868	**0.833**	0.819	0.721	0.767	0.798	0.85	0.823

From Table 4, we can see performances of the three classifier are similar. This demonstrates the features are well selected. NaiveBayes performance better in verb, adjective, and adverb synsets, while J48 work well for noun synsets. Accordingly, we use J48 to disambiguate noun synsets, and take NaiveBayese for the other ones. Table 4 give their results.

Table 5. Performance of classifier

	Noun	Verb	Adj	Adv	Average
Precision	82.14%	78.35%	81.22%	81.49%	81.37%
Coverage	86.71%	80.16%	83.91%	82.35%	85.21%
Average Words Number	4.13	6.25	6.00	3.01	4.62

4.3 *Lexicon-Syntactic Patterns Results*

Lexicon-syntactic patterns based disambiguation is time consuming. We did not take it as a classifier feature, but used as an heuristic rule. If one

pattern hits enough web pages, the candidate are accepted. Performance of this way is given in Table 6 with the former two ways.

Table 6. Performance of lexical patterns

	Clear synsetsx	Classifier	Lexical patterns
Precision	99.10%	81.37%	91.34%
Coverage	26.06%	47.21%	18.68%

5 Conclusion and Future Work

WordNet is an important resource for many applications but restricted to English, so translating it to Chinese is valuable. Our work is ground on the argument that concepts can be translated from one language to another expressed by synsets. The two major problems for the work are to translate English words and to choose the right translation for synsets. We firstly translate all the words in WordNet using three kinds of complementary resources, and then disambiguate the translation of synsets using a classifying combined with heuristic rules. Experiments show that our method can translate 85.12% of the synset in WordNet 3.0 with a precision of 81.37%.

Our future work will concentrate on how to improve the translate coverage of words, especially the multi-word expressions, in WordNet.

Acknowledgements This work was supported by the National Natural Science Foundation of China, under grants No. 61203284, 60573063, 60573064, 60773059, 61035004, the National High Technology Research and Development Program (863 Program) of China under No. 2007AA01 Z325, and National Social Science Foundation of China under grant No. 10AYY003.

References

1. Miller, G.A.: WordNet: A lexical database for English. Commun. ACM **38** (1995) 39–41
2. Global WordNet Association. `http://www.globalwordnet.org` (2000)

3. Piek, V.: EuroWordNet: A multilingual database with lexical semantic networks. Dordrecht: Kluwer Academic Publishers (1998)
4. Altangere, C., Ho-Seop, C., Cheol-Young, O., Hwa-Mook, Y.: On the evaluation of Korean WordNet. In: TSD 2007. (2007) 123–130
5. Mei, J., Zhu, Y., Gao, Y., , Yin, H.: TongYiCiCiLin. Shanghai Dictionary Press (1982)
6. Dong, Z., Dong, Q. `http://http://www.keenage.com` (2000)
7. Chen, H.H., Lin, C.C., Lin, W.C.: Construction of a Chinese-English WordNet and its application to CLIR. In: Proceedings of the Fifth International Workshop on Infor-mation Retrieval with Asian Languages. (2000)
8. Dorr, B.J., Levow, G.A., Lin, D.: Building a Chinese-English mapping between verb concepts for multilingual applications. In: Proceedings of 4th Conference of the Association for Machine Translation. (2000)
9. Chen, H.H., Lin, C.C., Wen, C.L.: Building a Chinese-English WordNet for translingual applications. ACM Transactions on Asian Language Information Processing **1**(2) (2002) 103–122
10. Liu, M.: A research on translating WordNet nodes to Chinese. Master's thesis, DongBei University (2003)
11. Green, R., Pearl, L., Dorr, B.J., Resnik, P.: Mapping lexical entries in a verbs database to WordNet senses. In: ACL 2001. (2001) 244–251
12. Huang, C.R., Tseng, I.J.E., Tsai, D.B.S.: Translating lexical semantic relations: The first step towards multilingual WordNets. In: COLONG 2002. (2002)
13. Fang, G., Yu, H., Nishino, F.: Chinese-english term translation mining based on semantic prediction. In: ACL 2006. (2006)
14. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press (1999)
15. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: COLING 1992. (1992) 539–545

SHI WANG
KEY LABORATORY OF INTELLIGENT INFORMATION PROCESSING,
INSTITUTE OF COMPUTING TECHNOLOGY,
CHINESE ACADEMY OF SCIENCES,
BEIJING, 100190, CHINA
E-MAIL: <WANGSHI@ICT.AC.CNX>

IJCLA VOL. 3, NO. 2, JUL-DEC 2012, PP. 77–92
RECEIVED 30/10/11 ACCEPTED 09/12/11 FINAL 08/06/12

Corpus Materials for Constructing Learner Corpus Compiling Speaking, Writing, Listening, and Reading Data

KATSUNORI KOTANI,[1] TAKEHIKO YOSHIMI,[2]
HIROAKI NANJO,[2] AND HITOSHI ISAHARA[3]

[1] *Kansai Gaidai University, Japan*
[2] *Ryukoku University, Japan*
[3] *Toyohashi University of Technology, Japan*

ABSTRACT

This paper presents the corpus material of a learner corpus called the I-Learner corpus consisting of text and sounds that reflect the proficiency of learners of English as a foreign language with respect to speaking, writing, reading, and listening, along with the types and quantity of the corpus materials. In constructing a learner corpus, a prerequisite is to prepare corpus materials that properly reveal learners' second language ability. Most conventional learner corpora use corpus materials taken from linguistic exercises such as essay writing and speaking exercises. The I-Learner corpus is the first corpus that collects the four-modality data, and the focus of this study is the selection of its material.

KEYWORDS: *Learner corpus, corpus materials, four-modality data*

1. INTRODUCTION

Learner corpora, which are defined as a collection of texts produced by learners of a second or foreign language [1], have contributed to the advancement of research on second language learning and teaching by providing text and sounds to analyze which linguistic items, such as vocabularies and grammars, learners adequately or inadequately use.

Some learner corpora [2, 3] are annotated with information tags on errors that learners made, thus making it possible to directly analyze learners' errors and/or compare the errors across learners of different proficiency levels. Learner corpora can also be used as a language resource in constructing computer-based language learning or teaching systems by machine learning algorithms [4].

The construction of a learner corpus consists of three steps: design, data collection, and analysis of collected data. The design step determines variables of a corpus. For example, the focus could be on language-related variables, task-related variables, and/or learner-related variables [5]. In the data collection step, raw text, sound, and information to be annotated with the text, such as learner information and error information, are collected. In the analysis of collected data step, basic analyses are performed, such as descriptive statistics analysis or qualitative analysis, to confirm the validity of the collected data.

Most learner corpora consist of text and sounds that reflect learners' proficiency in either writing [6] or speaking [2], but some include text that reflects learners' proficiency in the multiple modalities of speaking, writing, reading, and listening [7, 8, 9]. Wen et al. [7] constructed a learner corpus consisting of text that reflects learners' proficiency in speaking and writing. The speaking data included sounds and text transcribed from what learners had verbalized in speaking exercises, and the writing data included text from learners' essays. Meurers et al. [8] constructed a learner corpus consisting of text that reflects learners' reading and writing proficiency. The data included text written by learners as answers for comprehension questions in reading exercises. Kotani et al. [9] constructed a learner corpus, called the I(ntegrated)-Learner corpus, consisting of text and sounds that reflect learners' speaking (with a focus on pronunciation), writing, reading, and listening proficiency. According to them [9], one of the goals of this corpus is to provide a language resource for the analysis of learners' language use based on the four modalities because there is no other learner corpus that currently does so.

In constructing any learner corpus, the basic prerequisite is to select corpus materials that properly reveal learners' second language ability. Therefore, previous corpora have used materials taken from linguistic exercises such as essay writing [6, 7] and language tests [2, 7, 8, 9]. However, we feel that the selection of the corpus material of the I-Learner corpus [9] should be described in more detail because it is the first corpus that collects the four modality data. Therefore, in this paper

we discuss its design at length and also describe the types and quantity of the corpus materials.

2. I-LEARNER CORPUS

2.1. *Fundamental Design*

The I-learner corpus [9] was constructed on basis of the following design criteria: modality, context, technicality, data to be collected, learner, and task. In this subsection, we describe the modality, context, technicality, and data to be collected; the other criteria are described in the following subsections.

The modality consists of speaking, writing, listening, and reading. The context is the expository language used in daily-life contexts. The technicality is kept as low as possible in order to focus on linguistic proficiency. The data to be collected consist of language production data, language comprehension data, and mental language processing data.

The data to be collected are summarized in Table 1. The language production data, which show what the learners have produced, include both the sound of speaking and written sentences. The language comprehension data include the comprehension rate, which shows how well the learners comprehend the content of a text. The mental language processing data, which show how learners produced or understood sentences and/or sounds , include the speaking time, the writing time, the reading time, and the subjective judgment score, which is obtained by using a psychological data collection method [10] and shows what the learners thought as they were using English. The subjective judgment score of speaking on a five-point scale represents the difficulty of a sentence for the learner who pronounced that sentence. The subjective judgment score of writing on a five-point scale represents the comprehensibility of an English sentence written by a learner. The subjective judgment scores of listening and reading on a five-point scale represent the comprehensibility of a sentence for a learner who listened to or read the sentence.

Table 1. Data to be collected

	Language production data	Language comprehension data	Mental language processing data
Speaking	Sound	—	Speaking time Subjective judgment score
Writing	Sentence	—	Writing time Subjective judgment score
Listening	—	Comprehension rate	Subjective judgment score
Reading	—	Comprehension rate	Reading time Subjective judgment score

2.2. *Learners*

Learners of English as a foreign language were recruited, with candidates submitting their scores of the Test of English for International Communication (TOEIC) taken within a year of the start of the data collection. Ninety learners were accepted so as to obtain the same number of learners in each of the three proficiency levels: beginner (N = 30, TOEIC score of 280–495), intermediate (N = 30, TOEIC score of 500–725), and advanced (N = 30, TOEIC score of 730–985). The learners' first language was Japanese, and their education level was a university degree or higher, meaning that all had at least 36 months learning experience.

2.3. *Tasks of Data Collection*

The learners completed tasks (language tests of the four modalities) in the following order: listening, reading, speaking, and writing. For all tasks, they used a data collecting tool that displayed a sentence on a computer screen. This tool kept track of time when a learner verbalized, wrote, and read each sentence. It provided comprehension questions and saved answers for the listening and reading tasks. In the writing tasks, it displayed pictures and questions as well as blank spaces in which to write sentences. It kept a subjective judgment score during all the tasks.

In the listening tasks, the learners listened to four news articles that were read aloud by native speakers of English. They judged the difficulty of comprehending a sentence after listening to it. When they

finished listening to a news article, they answered five comprehension questions.

In the reading task, the learners silently read four news articles (which were different from the ones used in the listening task). They judged the difficulty of comprehending a sentence after reading it. When they finished reading a news article, they answered five comprehension questions. The use of a dictionary was prohibited, and the learners were allowed to read a sentence only once.

In the speaking task, the learners verbalized sentences from the four news articles that were used in the reading task. The same news articles were used so that the learners could grasp the content before the task began, thus enabling them to focus on pronunciation. They judged the difficulty of speaking a sentence after verbalizing it. There were no comprehension questions, unlike in the listening and reading tasks, because the focus was entirely on pronunciation, not comprehension.

In the writing task, the learners first described four pictures that comprised a series of events. They were assigned to write at least five sentences per picture. Next, they were provided with 20 questions, which they then answered. Here, they were assigned to write at least one sentence per answer. They judged the comprehensibility of a sentence after writing it. The use of a dictionary was prohibited, and the learners were not permitted to rewrite a sentence after they had moved on to another.

2.4. *Collected Data*

There were 90 learners who listened to 80 sentences from 4 news articles and answered 5 comprehension questions for each news article. Therefore, the listening data consisted of 7200 sentences annotated with a subjective judgment score and 360 examples of comprehension rate.

The reading data consisted of 7200 sentences annotated with the reading time and the subjective judgment score and 360 examples of comprehension rate. The total reading time was approximately 25.5 hours.

The speaking data consisted of 7200 sentences annotated with the speaking time and the subjective judgment score. The total speaking time was approximately 28.9 hours.

The 90 learners were asked to write at least 40 sentences for the writing task, so the writing data consisted of at least 3,600 sentences annotated with the writing time and the subjective judgment score. The

total writing time for the picture description was approximately 28.4 hours and that for answering questions was 30.2 hours.

3. MATERIALS OF I-LEARNER CORPUS

The materials used in the I-Learner corpus [9] were selected on basis of the design criteria (modality, context, technicality, data to be collected, learner, and task) described in Section 2.

3.1. *Material Design*

In compiling the learners' language data, we determined the design of corpus materials to emphasize the contrast between success and failure in that data. We designed the corpus materials to include three types of linguistic properties that enhance the contrast: the syntactic property of sentence length, semantic property of question type, and discourse property of information structure.

The speaking, listening, and reading materials were designed to include different syntactic difficulties and semantic difficulties. We used sentence length as an index of syntactic difficulties. Sentence length leads to difficulty in comprehending or processing linguistic objects, as previous research on readability [11] has shown. Thus, the news articles in the speaking, listening, and reading materials should contain different sentence lengths.

We used the type of question, such as true questions, false questions, and content questions, as an index of semantic difficulties. The effect of the type of question on the learners' language data should be examined in future work, but we expect that the question types cause the following differences in semantic difficulty. Content questions should be more difficult to answer than true questions and false questions because answers cannot be determined in a binary way (true or false). The language learners have to recognize what the article is about to answer content questions. In contrast, answers to true questions and false questions can be determined in a binary way. In addition, false questions should be more difficult than true questions to answer because deciding the correct answer to false questions, which needs negative evidence, requires more logical thinking than finding positive evidence.

The writing materials were designed to include different discourse difficulties and semantic difficulties. We used the discourse direction and the number of people in a picture [12] as an index of discourse difficulties. The effect of the discourse difficulties on the learners' language data should be examined in future work, but we expect that the discourse direction and the number of people in a picture cause the following difference in discourse difficulty. When describing these pictures, the learners have to represent the situation following the discourse direction on the basis of a proper information structure [13]. That is, when a new person appears, the person should be treated as new information. However, this person should be treated as old information in the subsequent picture. Thus, multiple pictures in the writing materials should represent a series of events, and different combinations of people should appear in each picture.

We used the type of question, such as polar or wh-interrogatives, as an index of semantic difficulties. The effect of the type of question on the learners' language data should be examined in future work, but we expect that the question types cause the following difference in semantic difficulty. Questions asking for descriptive comments should be the most difficult for which to write answers. The second most difficult should be wh-interrogative-type questions, and the least difficult should be polar-interrogative-type questions. Thus, questions in the writing materials should include these three types of questions.

3.2. *Speaking, Listening, and Reading Materials*

The speaking, listening, and reading materials of the I-Learner corpus were compiled from news articles taken from the Voice of America (VOA) site (http://www.voanews.com). The articles were chosen in two steps. In the first step, special sections for English learners and editorial sections were chosen from the various ones available on VOA. The articles in the former should be easier than those in the latter. This is because articles in special sections for English learners in VOA are written in short, simple sentences that contain only a core vocabulary of 1,500 words and no idiomatic expressions, according to VOA, while articles in editorial sections are written for native English speakers in sentences that have no restrictions. In the second step, articles were chosen according to conditions on the article size (number of words in an article) being approximately 350 words (within plus or minus 5%) and on the number of sentences in an article being 25 sentences for

easy articles and 15 sentences for difficult articles. These conditions excluded the possibilities that easy articles contained more long sentences and that difficult articles contain more short sentences.

The same articles are used when compiling the speaking and reading data. First, the learners silently read four articles (two easy and two difficult ones), and then they read aloud those same articles. The first reading enables the learners to grasp the content of the articles. Thus, when reading aloud, they can focus on the pronunciation. Examples of an easy and a difficult article, respectively, are shown in Appendices 1 and 2. When reading an article silently or aloud, the learners see this article on a computer screen sentence by sentence.

The listening data are also compiled using four articles (two easy and two difficult ones). These articles were taken from the same sections of the VOA site as those used in the speaking and reading tasks. In addition, these articles met the conditions for the article size and number of sentences in an article. In the listening task, the learners listen to VOA reporters.

The linguistic properties of the articles used in the speaking and reading tasks are shown in Table 2, and the properties of the articles in the listening task are shown in Table 3. These tables provide the difficulty of the article (Difficulty: Easy or Difficult), the title of the article (Title), the number of words in an article (W), the number of words in the shortest sentence (Min), the number of words in the longest sentence (Max), the average number of words in the sentences (Mean), and the standard deviation (SD).

A one-way analysis of variance (ANOVA) was conducted to examine whether the sentence length (number of words per sentence), as an index for syntactic difficulties, differed between the easy and difficult articles. The article difficulty was determined based on the type of sections: special sections for English learners or editorial sections for native English speakers. There was a significant difference in the sentence length at the $p<.01$ level [$F(3, 76)=14.16$] in the articles for the speaking and reading tasks. Post-hoc comparisons using the Tukey honestly significant difference (HSD) test indicated that the mean values of the sentence lengths were significantly different between all the pairs of easy articles (E1, E2) and difficult articles (D1, D2). However, there was no significant difference between E1 and E2, or between D1 and D2.

Table 2. Properties of speaking and reading materials

Article ID	E1	E2	D1	D2
Difficulty	Easy	Easy	Difficult	Difficult
Title	Recruiters Help US Colleges Find Foreign Students	Book Predicts Jump in High School Courses Online	U.S. Designates Al-Quso Terrorist	Ending Impunity In the Congo
W	337	356	359	348
Min	7	5	12	11
Max	23	22	37	42
Mean	13.5	14.2	23.9	23.2
SD	4.6	4.2	7.7	10.1

Table 3. Properties of listening materials

Article ID	E3	E4	D3	D4
Difficulty	Easy	Easy	Difficult	Difficult
Title	Studying in the US: A Lesson in Personal Finance, Part 2	Studying in the US: Grading Grades	Educating Marginalized Children	Outreach To Muslims
W	358	341	357	353
Min	5	6	8	10
Max	22	20	39	38
Mean	14.3	13.6	23.8	23.5
SD	4.8	3.7	8.9	7.4

There was also a significant difference in the sentence length at the $p < 0.01$ level [$F(3, 76) = 16.22$] in the articles for the listening task. Post-hoc comparisons using the Tukey HSD test indicated that the mean values of the sentence lengths were significantly different between all the pairs of easy articles (E3, E4) and difficult articles (D3, D4). However, there was no significant difference between E3 and E4, or between D3 and D4. Taken together, these results show that the easy articles contain shorter sentences than the difficult articles.

The listening and reading materials included questions created by the author of this paper following question formats [14]. The questions

are categorized into three types: a question asking what is true, e.g., "Which of the following is mentioned?" (true question); what is false, e.g., "Which of the following is NOT mentioned?" (false question); and what the content is about, e.g., "According to the passage, why or how…?" (content question). Each article has two true questions, two false questions, and one content question. Appendix 3 illustrates the questions for the easy article shown in Appendix 1. The questions are multiple choices with four answer choices.

3.3. *Writing Materials*

In the picture description task, the learners describe a series of events. The events are represented in a series of four pictures (Appendix 4), and thus this material represents the discourse direction. Four people appear in these pictures. In picture A, a woman and a man appear. In picture B, a different man appears with the woman and man who appeared in picture A, for a total of three people. In picture C, only the two men appear. In picture D, a different woman appears with the other three people.

Given the discourse difficulties of the order of pictures and the number of people, describing picture D should be most difficult. The second-most difficult picture should be picture B or C. If the order of pictures contributes more to the difficulty of describing pictures, the difficulty of picture C would be greater than that of picture B. In contrast, if the number of people has a greater effect on the difficulty of describing pictures, picture B would be more difficult than picture C.

In the question answering tasks, the learners answer questions about their own learning profiles [15] and on their computer literacy [16] (Appendix 5). The sentences from 1 to 15 ask about the learners' learning profiles, and those from 16 to 20 ask about their computer literacy. Of these sentences, 13 are wh-interrogative-type and 5 are polar-interrogative-type questions. The remaining two sentences are not interrogatives; instead, they ask for descriptive comments.

4. CONCLUSION

The present paper introduced the corpus materials of the I-Learner corpus, which collected learners' language data for the four modalities of speaking, writing, listening, and reading. These materials were

designed to include different linguistic difficulties. The writing materials included different semantic difficulties and discourse difficulties: the type of question, the discourse direction, and the number of people in a situation. The speaking, listening, and reading materials included different semantic difficulties and syntactic difficulties: the type of question and the sentence length.

We further noted the expected effects of these linguistic difficulties on the learners' language data. However, we have not examined whether these effects appear in that data. This examination will provide fundamental information for assessing the validity of the corpus for future studies. Thus, one remaining issue is to examine whether the corpus materials actually emphasize the contrast between success and failure in learners' language data after compiling the relevant data.

REFERENCES

1. Tono, Y.: Integrating Learner Corpus Analysis into a Probabilistic Model of Second Language Acquisition. In P. Baker (ed.) Contemporary Corpus Linguistics. Continuum International Publishing Group, London, pp. 184–203 (2009).
2. Izumi, E., Uchimoto, K., Isahara, H. (eds.): Nihonjin 1200 Nin no Eigo Spiking Koopasu [A Speaking Corpus of 1200 Japanese Learners of English]. ALC Press, Tokyo, Japan (2004).
3. Gammon, M.: High-order Sequence Modeling for Language Learner Error Detection. Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications, pp. 180—189, (2011).
4. Kotani, K., Yoshimi, T., Kutsumi, T., Sata, I., Isahara, H.: EFL Learner Reading Time Model for Evaluating Reading Proficiency. CICLing 2008, pp. 655–664 (2008).
5. Tono, Y.: Learner Corpora: Design, Development and Applications. Paper presented at the Corpus Linguistics 2003 Conference (CL 2003), (2003).
6. Granger, S., Dagneaux, E., Meunier, F., Paquot, M.: International Corpus of Learner English, version 2. Presses Universitaires de Louvain, Louvain-la-Neuve, Belgium, (2009).
7. Wen, Q., Liang, M., Yan, X.: Spoken and Written Corpus of Chinese Learners (SWECCL) 2.0. Foreign Language Teaching and Research Press, Beijing, China, (2008).
8. Meurers, D., Ott, N., Ziai, R.: Compiling a Task-based Corpus for the Analysis of Learner Language in Context. In Sam Featherston and Britta Stolterfoht, editors, Proceedings of Linguistic Evidence 2010, pp. 214–217, (2010).

9. Kotani, K., Yoshimi, T.: A Scoring Method for Second Language Writing based on Word Alignment. Proceedings of Pacific Association for Computational Linguistics (PACLING) 2011, (2011).
10. Lewis, C. H.: "Thinking Aloud" Method in Cognitive Interface Design. Technical Report IBM RC-9265, (1982).
11. Kate, R. J., Luo, X., Patwardhan, S., ranz, M., Florian, R., Mooney,R. J., Roukos, S., Welty, C.: Learning to Predict Readability Using Diverse Linguistic Features. Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pp. 546–554, (2010).
12. McCarthy, M.: Discourse Analysis for Language Teachers. Cambridge University Press, Cambridge, (1991).
13. Prince, E. F.: Toward a Taxonomy of Given-new Information. In Peter Cole, editor, Radical Pragmatics, Academic Press, New York, pp. 223–255, (1981).
14. Nation, P., Malarcher, C.: Reading for Speed and Fluency. Compass Publishing, Seoul, Korea, (2007).
15. Ehrman, M. E.: Understanding Second Language Learning Difficulties. SAGE Publications, London, (1996).
16. Eignor, D., Taylor, C., Kirsch, I., Jamieson, J.: Development of a Scale for Assessing the Level of Computer Familiarity of TOEFL Examinees. Research Reports RR98-7, Educational Testing Service, Princeton, New Jersey, (1998).

APPENDICES

Appendix 1. Easy Article in Speaking and Reading Tasks

01: College prices in the United States have been rising faster than other prices for thirty years or more.
02: Recently many of the nation's top colleges have agreed to increase their financial aid.
03: But one group often has to pay the full price for college: foreign students.
04: This may help explain why colleges are making greater efforts to recruit them.
05: Large universities are likely to use their own representatives.
06: But smaller schools may work with independent recruiters.
07: An example is Albright College in Reading, Pennsylvania.
08: It has about one hundred foreign students, mostly from Asia.
09: It offers foreign students a savings of one-fifth off its published price if they apply through Study Group Holdings.
10: This placement company operates the Web site go-study.com.
11: Albright's international student counselor, Nicole Christie, says the company is paid from the money that the students pay the college.

12: Study Group looks for qualified students and rates their English skills before they apply.
13: But foreign students themselves often pay recruiters.
14: The recruiters help them write applications, get recommendation letters and prepare for admissions tests.
15: And they might help students prepare for getting a visa to study in the United States.
16: Recruiters can also work for both students and colleges.
17: Some education officials call this a conflict of interest.
18: They wonder how recruiters can find a school that is truly right for a student when certain colleges are paying them.
19: Officials also warn that like any other business, there is a risk of dishonesty.
20: Recruiters say they provide a useful service that is legal in the United States.
21: They say the colleges they work for are accredited and provide a good education but may not be widely known.
22: Recruiting of foreign students has been the subject of recent stories in the Chronicle of Higher Education and in the New York Times.
23: We are interested in hearing about experiences with college recruiters.
24: Send us your comments and we may use them in a future report.
25: Write to special@voanews.com and please include your name and country.

Appendix 2. Difficult Article in Speaking and Reading Tasks

01: The United States and the United Nations have listed Al-Qaida in the Arabian Peninsula fugitive Fahd al-Quso as a Specially Designated Terrorist.
02: These actions will help stem the flow of finances to and inhibit the travel of this dangerous operative.
03: The designation of Fahd al-Quso highlights U.S. action against the threat posed to the United States by al-Qaida in the Arabian Peninsula, said U.S. Ambassador for Counterterrorism Daniel Benjamin.
04: The joint designation by the United States and the United Nations alerts the public that Fahd al-Quso is actively engaged in terrorism.
05: These actions, said Ambassador Benjamin, "expose and isolate individuals like al-Quso and result in denial of access to the global financial system."
06: Prior to the formation of al-Qaida in the Arabian Peninsula, or AQAP, al-Quso was associated with al-Qaida elements in Yemen and involved in the 2002 USS Cole bombing in the Port of Aden, which killed seventeen sailors.
07: He was jailed in Yemen in 2002 for his part in the attack.
08: Following al-Quso's release from prison in 2007, he joined al-Qaida in Yemen.
09: In November 2009, al-Quso was added to the list of the FBI's most wanted terrorists.
10: Al-Quso is connected to other designated AQAP senior leaders, including Anwar al-Awlaqi, Nasir al-Wahishi, and Said Ali al-Shiri, and acts as a cell leader in Yemen.

11: In May 2010, al-Quso appeared in an al-Qaida in the Arabian Peninsula video in which he threatened to attack the U.S. homeland, as well as U.S. embassies and naval vessels abroad.
12: The terrorist designation blocks all al-Quso's property interests subject to U.S. jurisdiction and prohibits U.S. citizens from engaging in transactions that benefit al-Quso.
13: In addition to the U.S. domestic action, the United Nations Sanctions Committee's listing will require all U.N member states to implement an assets freeze, a travel ban, and an arms embargo against al-Quso.
14: The actions taken against the AQAP operative demonstrate international resolve in eliminating its ability to execute violent attacks and to disrupt, dismantle, and defeat their networks.
15: This designation represents just one phase of the U.S. government's response to the threat posed by al-Qaida in the Arabian Peninsula.

Appendix 3. Comprehension Questions for Easy Article Shown in Appendix 1

1. Which of the following is mentioned?
   (a) College teams from around the world took part in a computer programming competition.
   (b) Second of two reports on the business of bringing together students and schools.
   (c) Wealthier countries agree to limit how aggressively they recruit from developing countries.
   (d) Placement companies may be paid by colleges or students -- or both, raising concerns about possible conflicts of interest.

2. Which of the following is mentioned?
   (a) Universities will make greater efforts to recruit foreign students.
   (b) Universities agreed to increase their financial aid for foreign students.
   (c) Universities operate the Web site go-study.com.
   (d) Universities are interested in hearing about experiences with college recruiters.

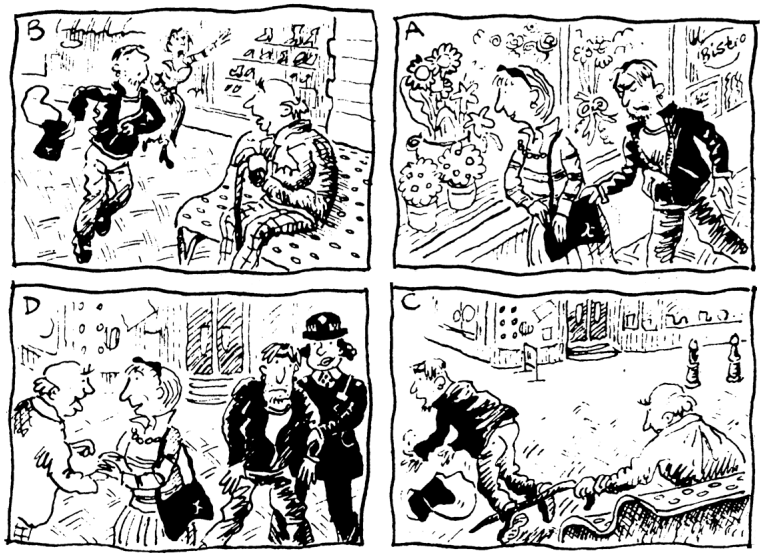
3. According to the passage, why do universities make efforts to recruit foreign students?
   (a) Because college prices have been rising.
   (b) Because universities work with independent recruiters.
   (c) Because foreign students have to pay the full price for college.
   (d) Because universities look for qualified students.

4. Which of the following is NOT mentioned?
   (a) A college offers foreign students a savings of one-fifth off its published price.
   (b) Recruiters help foreign students prepare for admissions tests.

(c) Recruiters work for both students and colleges.
(d) Large universities work with independent recruiters.

5. Which of the following is NOT mentioned?
   (a) Recruiters provide a useful service that is illegal in the United States.
   (b) Recruiters help foreign students prepare for getting a visa to study in the United States.
   (c) Some colleges providing a good education may not be widely known.
   (d) You can send them your comments.

Appendix 4. Pictures for Description



Appendix 5. Sentences for Question Answering

1. What were your favorite subjects?
2. What were your least favorite subjects?
3. What were your TOEIC scores (most recent)?
4. When did you last attend a class or take a course of any sort?
5. What was the class?
6. Which languages do you speak and read, and how well?
7. What language did you learn?
8. How did you learn the language?
9. How long did you learn the language?

10. Did you enjoy it?
11. Were you ever in contact with other languages while growing up? If yes, please describe briefly.
12. Did you find learning foreign languages easy?
13. Is there anything that might interfere with your learning and using another language?
14. Please add any additional comments about your past or anticipated language learning experience that might be helpful.
15. A variety of techniques may be used to help you learn foreign languages, by you and by your teachers. Please describe them.
16. How often is there a computer available for you to use at home?
17. How comfortable are you with using a computer?
18. How comfortable are you with using a computer to write a paper?
19. How many examinations/tests have you taken on a computer?
20. How often do you use a computer to send or receive e-mail?

KATSUNORI KOTANI
KANSAI GAIDAI UNIVERSITY,
16-1 NAKAMIYAHIGASHINO-CHO,
HIRAKATA, OSAKA, 573-1001, JAPAN
E-MAIL: <KKOTANI@KANSAIGAIDAI.AC.JP>

TAKEHIKO YOSHIMI
RYUKOKU UNIVERSITY,
1-5 YOKOYA SETA OE-CHO,
OTSU, SHIGA, 520-2194, JAPAN

HIROAKI NANJO
RYUKOKU UNIVERSITY,
1-5 YOKOYA SETA OE-CHO,
OTSU, SHIGA, 520-2194, JAPAN

HITOSHI ISAHARA
TOYOHASHI UNIVERSITY OF TECHNOLOGY,
1-1 HIBARIGAOKA, TEMPAKU,
1-2 TOYOHASHI, AICHI, 441-8580, JAPAN

IJCLA VOL. 3, NO. 2, JUL-DEC 2012, PP. 93–104
RECEIVED 04/11/11 ACCEPTED 09/12/11 FINAL 01/10/12

Using the ILCI Annotation Tool for POS Annotation: A Case of Hindi

RITESH KUMAR, SHIV KAUSHIK, PINKEY NAINWANI, ESHA BANERJEE, SUMEDH HADKE, GIRISH NATH JHA

Jawaharlal Nehru University, New Delhi

ABSTRACT

In the present paper, we present an annotation tool, ILCIANN (Indian Languages Corpora Initiative Annotation Tool), which could be potentially used for crowd-sourcing the annotation task and creation of language resources for use in NLP. This tool is expected to be especially helpful in creating annotated corpora for the less-resourced languages. ILCIANN is a server-based web application which could be used for any kind of word-level annotation task in any language. In the paper a description of the architecture of the tool, its functionality, its application in the ILCI (Indian Languages Corpora Initiative) project for POS annotation of Hindi data and the extent to which it increases the efficiency and accuracy of the annotators is given. It describes the results of an experiment conducted to understand the increase in the efficiency (in terms of time spent on annotation) and the reliability (in terms of the inter-annotator agreement) with the use of the tool when compared to the manual annotation.

KEYWORDS: ILCIANN, ILCI, POS annotation, server-based annotation, Hindi POS annotation

1 INTRODUCTION

ILCIANN is a server-based web application which could be used for any kind of word-level annotation task in any language. It is developed using Java/JSP as the programming language and is running on Apache

Tomcat 4.0 web server. It is meant to facilitate the job of manual annotation (and not be a tagger in itself) by providing a user interface. It also provides the facility of limited automatic tagging for closed grammatical categories like pronouns, postpositions, conjunctions and quantifiers which reduces the burden of human annotators.

Some other annotation tools have been developed for similar purposes. Bird et al. [6] came up with a tool which targeted at facilitating the development of linguistic annotations called Atlas (Flexible and Extensible Architecture for Linguistic Annotations). It consists of three levels:

1. The logical level: defines a set of procedures for creating, modifying, searching, and storing well-formed annotation sets
2. The physical level: free to access in various ways- via networked client server modes , or via linked libraries into application binaries, or via scripting languages
3. The application level: reduces the burden of human annotators and also language engineering application development.

Though the tool is comprehensive in nature but it works best for speech database and corpus.

Kaplan et al. [7] designed a web based annotation tool (SLATE: Segment and Link-based Annotation Tool Enhanced), which addresses ten major annotation needs:

1. Managing the role of annotator and administrator,
2. Delegation and monitoring work,
3. Adaptability to new annotation tasks,
4. Adaptability within the current annotation task,
5. Diffing and merging (diffing and merging of data from multiple annotators on a single resource to create a gold standard),
6. Versioning of corpora,
7. Extensibility in terms of layering,
8. Extensibility in terms of tools,
9. Extensibility in terms of importing/exporting and,
10. Support for multiple languages.

This tool to a great extent addresses to the purpose of the management of large and parallel data but it does not address the issue of the annotation of translated parallel corpora.

2. THE ILCIANN TOOL

The tool is being developed and currently used for POS annotation in the Indian Languages Corpora Initiative (ILCI) project funded by the Department of Information Technology (DIT), Govt. of India ([3, 4]). The first phase of the project involved developing a POS annotated parallel translated corpus of 50,000 sentences in 12 major Indian languages (which included Hindi, Urdu, Bangla, Oriya, Punjabi, Gujarati, Marathi, Konkani, Telugu, Tamil, Malayalam and English). It is a consortium project running parallel in 10 different universities of India spread across the country. The basic corpus was prepared in Hindi, which was translated in 10 other languages to prepare the parallel corpus. Once the corpus creation was complete, the data had to be annotated with labels for part of speech (POS) using the BIS tagset (a newly framed tagset, approved by Bureau of Indian Standards (BIS), which is now the national standard and supposed to be used in all kinds of POS annotation work across the nation).

In order to manage the whole process of annotation in such a way that it could be done efficiently and with minimum errors, the ILCI Annotation Tool (ILCIANN) is being used. The use of the tool ensured that the data is saved in a centralized server in a uniform format which could be later utilized for any NLP task without much need of pre-processing or noise cleaning.

The following sections describe the architecture and working of the tool.

2.1 *Architecture of the Tool*

2.1.1 *Module 1 (Admin Module)*

This is the module where all the administrative work related to any annotation project is carried out. The following steps are carried out in this module (and they are the most basic steps that need to be taken before starting any annotation project and during the project also)

1. **Step 1 (Creating the user login)**: This step involves creating the login of users who would annotate the data. The project administrator has the authority to create the login for the number of specific human annotators who want to annotate/tag the data. It

ensures the safety as well as authenticity of the tagged data, while theoretically giving an opportunity to a huge community to support and help in building language resources for their language. Moreover if the annotation project involves more than one language then the user is also assigned the language on which (s)he is supposed to work. For instance, if x is Hindi language annotator in a multi-language project, (s)he can only work on Hindi data and cannot do any modification (tagging the data, editing the data and saving it) in other language files. Furthermore, each user is assigned a set of maximum 3 files for annotation at one time (and a new file is assigned only after one of the files is completed) to ensure that multiple users do not work on same file (which also helps in keeping a record of the progress of the individual annotators) and also that one or more files are not left incomplete.

2. **Step 2 (Uploading the Files)**: This step involves uploading various files which would be used for the annotation and include the data files which need to be tagged, the tagset which is to be used and a file called the *autotag* file. The autotag file consists of a list of words (which belong to closed grammatical category) and their POS label. This file is used by the tool to tag the function words automatically.
3. **Step 3 (Monitoring the Progress)**: The admin could also monitor the progress of each and every user in his/her project. The information includes the number of files completed by each user, the name of the files assigned to each user, the files on which each user is currently working, etc.
4. **Step 4 (Downloading the Files)**: The file is ready for download only when each sentence of the file is tagged. Downloading the completed file is optional and only the administrator of the project has the right to download these files.

2.1.2 *Module 2 (Annotation Module)*

1. **Step 1 (Selection)**: After the user logged in, the left hand side of the page shows two options: select the file and sentence id. The user is required to select the file in which (s)he wants to work. Once the file gets selected, the untagged sentence immediately appears. . Further, if the user wants to do some modifications in previously tagged sentences, (s)he can do it with the option of “select a sentence id”. The right hand side of the page shows the

progress of tagging status i.e. number of completed tagged sentences and also completed files.

2. **Step 2 (Editing/Segmentation)**: This step is optional. The user uses this button only when there is some error in the original data which needs to be corrected.
3. **Step 3 (Annotation):** This is the major step in the tool. As "tag the sentence" button is clicked, each word of the sentence with the default tag (the first tag in the tagset) appears except for the words which are automatically tagged. As mentioned above, to minimize the human efforts, the ILCIANN tool automatically tags closed categories like pronouns, postpositions, quantifiers, symbols and punctuations. These automatically tagged words are not frozen, as we know that part-of-speech is purely contextual, therefore, one may want to do further modifications on automatically tagged words also if (s)he finds it inappropriate according to the context, (s)he has the option to do so. Words which are not tagged, the user selects the appropriate tag from the given tagset list.
4. **Step 4 (Saving)**: After assigning the appropriate tag to each word, there is the button of "save" which saves the tagged sentence. The whole file cannot be saved in one go, each and every sentence needs to be saved individually. The saved tagged sentence is stored on the server in the format of "sentence id" and respective "tagged sentence".

2.1.3. *Module 3 (Statistics Module)*

1. **Information 1 (File Information):** This includes information regarding the number of files completed and the number of files on which work is in process.
2. **Information 2 (Sentence Information)**: The information regarding the number of sentences completed in the present file and in the whole corpus, and also the speed of annotation of each user (in terms of sentences per minute) is included here.

2.2 *Using the Tool: POS Annotation in ILCI*

There are three levels of users of this tool:

1. **Administrator (Admin)**: For the purpose of management, each language is assigned an administrator user account or the Admin

account. The Admin has a username and password, which he or she uses to access his/her account. It is in the Admin's jurisdiction to assign annotation work to as many Users as is required, the language in which annotation work will be carried out as well as up to 3 sets within each language group. The tasks of the Admin include maintaining the log of user details, tagging status and downloading completed files.

2. **User**: The User is assigned a username, password and language. The User, on entering this information in the Login page is directed to the main Home page of the tool, wherein the sets that he or she is assigned are displayed. The User selects the set number and the sentence ID which (s)he wants to work on. In case there is a need for correction within the displayed sentence, the User uses the Segment button to insert or delete additional information, such as white space removal, hyphen insertion etc. Once the sentence is ready for tagging, the User clicks on Tag the Sentence button. On clicking the button, each word of the sentence is displayed separately with the tagset in a drop-down box format beside each word. The User selects the appropriate tag for each word and tags the sentence. On completion, the sentences, along with the tags, are saved with the help of Save button. On completion of work, the User logs out using the Logout button.
3. **Master Admin**: The Master Admin also has a Username and password, which he or she uses to access his/her account. In addition to the normal tasks of the Admin, the Master Admin can also maintain the time log of the user accounts and create, delete, or change passwords of user accounts.

3 EFFICIENCY AND RELIABILITY OF THE TOOL

In order to understand the efficiency and reliability of the tool, an experiment was conducted with the help of three annotators. Each annotator was given two sets of data, each containing around 500 words (a total of 45 sentences). These sentences were taken from the ILCI corpora and contained almost equal number of words from both the health and tourism domain. The annotators were required to annotate the words manually (in a text file, without using any kind of tool), using the tool without intelligence and using the tool with intelligence. While the first set of 500 words were same across all these methods of annotation, the second set of 500 words were different

across all these methods. As is common practice in such experiments, the annotators were not allowed to consult each other during the annotation period. The experiments were conducted over a period of 6 days, with a gap of one day in between the annotation by each method (to reduce the bias in the common set). The time taken by each annotator in annotating each set by each method was noted down. Also the tagged data is being used to calculate the inter-annotator agreement in order to see if the tool also increases the reliability of the annotation process.

3.1 *Calculating the Efficiency*

Table 1 gives the time taken by each annotator in annotating each set by each method.

Table 1. Comparison of time taken in annotation (in minutes)

	Manual		Not intelligent		Intelligent	
Sets	A	B	A	B	A	B
Annotator A	55	50	30	35	15	15
Annotator B	32	36	22	25	18	17
Annotator C	125	97	29	33	24	16

As we could clearly see the tool (without any intelligence) has led to almost 100% increase in the efficiency of annotator A (for set A). While for others also there is an increase of around 50% in the efficiency of annotator A and B. While for annotator C, we see that the speed (which was very slow when the annotation was carried out manually) has increased tremendously and has come at par with the other two annotators. Moreover when we impart some intelligence to the system, we again see an increase of almost 50% in the efficiency of annotator A; while there is a marked increase in the speed of other annotators also. This efficiency could be further increased by imparting more intelligence to the machine. It must be noted that the intelligence, at present, is given to the machine by way of an *autotag* file which consists of a list of word with the tag that should be given to it. This file is prepared manually and contains those words which always takes only one tag irrespective of the context (mainly function words; but it also has some content words). At a later stage the tool will be equipped

with machine learning algorithms so that it becomes a POS tagger in effect and it could auto-tag most of the words and the user's effort remains only in revising the annotated data.

3.2 *Calculating the Reliability*

Several methods (discussed in detail) are used to compute the reliability (or, inter-annotator agreement) of any annotation work. Some of the major ones include the following.

Percentage Agreement (also called observed agreement, defined by Scott, 1955) is one of the simplest and earliest measures of inter-annotator agreement where the percentage of agreements between two annotators is calculated.

Cohen's kappa coefficient [1] is one of the best-known statistical measures of inter-rater agreement or *inter-annotator agreement* (IAA) for qualitative items. It is generally thought to be a more robust measure than simple percent agreement calculation since K takes into account the agreement occurring by chance. Cohen's kappa measures the agreement between two raters and each classifies N items into C mutually exclusive categories. The equation for K is

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)},$$

where Pr(a) is the relative observed agreement among raters, and Pr(e) is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly saying each category. If the raters are in complete agreement then K = 1. If there is no agreement among the raters other than what would be expected by chance, K = 0.

Scott's pi [5] is a statistic for measuring inter-rater reliability for nominal data. Scott's pi is similar to Cohen's kappa in that they improve on simple observed agreement by factoring in the extent of agreement that might be expected by chance. On the other hand Scott's pi makes the assumption that annotators have the same distribution of responses, which makes Cohen's kappa slightly more informative. The equation for Scott's pi, as in Cohen's kappa is:

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)};$$

however, Pr(e) is calculated using joint proportions.

Fleiss' Kappa [2] is a generalization of Scott's pi statistic. It is a statistical measure for assessing the reliability of agreement between a fixed number of raters when assigning categorical ratings to a number of items or classifying items. It works for any number of raters giving categorical ratings to a fixed number of items unlike Cohen's kappa and Scott's pi. It can be interpreted as expressing the extent to which the observed amount of agreement among raters exceeds what would be expected if all raters made their ratings completely randomly. Fleiss' kappa specifically assumes although there are a fixed number of raters (e.g., three), different items are rated by different individuals (Fleiss, 1971, p.378). If a fixed number of people assign numerical ratings to a number of items then the kappa will give a measure for how consistent the ratings are. The kappa, K, can be defined as:

$$\kappa = \frac{\overline{P} - \overline{P}_e}{1 - \overline{P}_e}.$$

The factor gives the degree of agreement that is attainable above chance, and $\overline{P} - \overline{P}_e$ gives the degree of agreement actually achieved above chance. If the raters are in complete agreement then K = 1. If there is no agreement among the raters then K = 0.

For the present purposes, Cohen's Kappa and Scott's Pi are not very relevant since the experiment involved more than two annotators. However we have calculated both the percentage and the Fleiss' Kappa so that the agreement measure of both kinds (taking chance into account and without taking chance into account) is calculated.

3.3 *Calculating Percentage Agreement*

The simple percentage of agreements among the three pairs of annotators is summarised in Table 2. It is calculated using the simple formula of percentage: sum of agreed instances × 100 / total number of instances.

While the inter-annotator agreement between annotators A and B is already on the higher side of the spectrum, it does not improve much with the use of the tool and it seems that the other factors (like the tagset itself, the guidelines, annotators' expertise, etc.) are playing a vital role here. However the situation is quite different in case of

agreement between annotators B and C and that between A and C where the inter-annotator agreement in case of manual annotation is pretty low. The agreement between the annotators improves quite considerably with the use of the tool. The intelligence of the tool also seems to be playing some role in the improvement of the inter-annotator agreement.

Table 2. Percentage of agreement among three pairs of annotators (%)

	Manual		Not intelligent		Intelligent	
Sets	A	B	A	B	A	B
Annotators A and B	85	87	84	83	90	87
Annotators B and C	66	77	81	81	83	85
Annotators A and C	67	72	76	81	81	80

3.4 *Calculating Fleiss' Kappa*

As mentioned earlier Fleiss' Kappa is a generalization over Scott's pi to calculate the inter-annotator agreement among more than 2 annotators. Since the present experiment involved three annotators, Fleiss' Kappa was also calculated (which is generally considered more reliable and accurate than percentage calculation). In order to arrive at a better picture vis-a-vis the percentage agreement as well as see if the overall agreement is affected by one annotator, both the inter-annotator agreement in between each pair of annotators as well as the overall agreement is also estimated. The values of Fleiss' Kappa for each pair of annotator in each set and also the general values for all the sets taken together is summarised in Table 3.

These values of Fleiss' reaffirm the facts that were shown by the percentage calculation of the agreements. The tool seems to be making only a small contribution to an increase in the reliability of the annotation at the present stage. However when we look at the overall result, we see a steady increase in the reliability (or, inter-annotator agreement) of the annotation efforts as we move from manual annotation to annotation using the tool to annotation using the tool with some limited intelligence.

Table 3. Calculated values of Fleiss' Kappa

Annotators	Manual		Not intelligent		Intelligent	
Sets:	A	B	A	B	A	B
A and B	0.852	0.871	0.829	0.820	0.895	0.881
B and C	0.698	0.786	0.794	0.796	0.814	0.867
A and C	0.719	0.732	0.731	0.803	0.789	0.819
A, B and C	0.757	0.797	0.785	0.806	0.833	0.856
A, B and C	0.777		0.797		0.845	

4 CONCLUSIONS

In the present paper, we have described the working of an online annotation tool, ILCIANN, which is meant not only to facilitate the task of manually annotating the data but also increase the overall efficiency (by considerably reducing the time taken in the annotation work) and the reliability (by increasing the inter-annotator agreement) of the annotation task. The experiments conducted to know the exact nature of efficiency and reliability has clearly shown that both of these attributes increase as the intelligence of the tool increases. Since the tool is developed in such a way that it could become more intelligent as more annotation takes place, the tool is expected to work in a much better way as the time passes and it could prove to be a very useful resource for the development of language resources for all kinds of language, especially the less-resourced ones.

REFERENCES

1. Cohen, J.: A coefficient of agreement for nominal scales. Educational and Psychological Measurement. 20(1), 37–46 (1960)
2. Fleiss, J. L.: Measuring nominal scale agreement among many raters. Psychological Bulletin. 76(5), 378–382 (1971)
3. Jha, G. N.: Indian Language Corpora Initiative (ILCI). Invited talk, 4th Intern. Language and Technology Conf. (4th LTC), Poland (2009).
4. Jha, G. N.: The TDIL program and the Indian Language Corpora Initiative (ILCI). In: Proceedings of the Seventh International conference on Language Resources and Evaluation (LREC'10), pp. 982-985 (2010)
5. Scott, W.: Reliability of content analysis: The case of nominal scale coding. Public Opinion Quarterly. 19(3), 321–325 (1955)
6. Bird, S., David D., Garofolo, J. S., Henderson, J., Laprun, C., Liberman, M.: Atlas: A flexible and extensible architecture for linguistic annotation. CoRR, cs.CL/0007022 (2000)

7. Kaplan, D., Iida, R., Tok, T.: Annotation Process Management Revisited. In: Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), pp. 3654-3661 (2010)

RITESH KUMAR
CENTRE FOR LINGUISTICS,
JAWAHARLAL NEHRU UNIVERSITY,
NEW DELHI, INDIA.
E-MAIL: <RITESH78_LLH@JNU.AC.IN>

SHIV KAUSHIK
SPECIAL CENTRE FOR SANSKRIT STUDIES,
JAWAHARLAL NEHRU UNIVERSITY,
NEW DELHI, INDIA
E-MAIL: <SHIVKAUSHIK.ENGG@GMAIL.COM>

PINKEY NAINWANI
CENTRE FOR LINGUISTICS,
JAWAHARLAL NEHRU UNIVERSITY,
NEW DELHI, INDIA
E-MAIL: <PINKEYBHU39@GMAIL.COM>

ESHA BANERJEE
CENTRE FOR LINGUISTICS,
JAWAHARLAL NEHRU UNIVERSITY,
NEW DELHI, INDIA
E-MAIL: <ESHA.JNU@GMAIL.COM>

SUMEDH HADKE
CENTRE FOR INDIAN LANGUAGES,
JAWAHARLAL NEHRU UNIVERSITY,
NEW DELHI, INDIA
E-MAIL: <SUMEDHKHADKE@GMAIL.COM>

GIRISH NATH JHA
SPECIAL CENTRE FOR SANSKRIT STUDIES,
JAWAHARLAL NEHRU UNIVERSITY,
NEW DELHI, INDIA
E-MAIL: <GIRISHJHA@GMAIL.COM>

Parsing and Co-reference

IJCLA VOL. 3, NO. 2, JUL-DEC 2012, PP. 107–122
RECEIVED 31/10/11 ACCEPTED 09/12/11 FINAL 21/12/12

POS Taggers and Dependency Parsing

RAMADAN ALFARED AND DENIS BÉCHET

University of Nantes, France

ABSTRACT

A wide-coverage parser copes with the problem of the explosion of the number of combinations of sub-trees and the number of theoretically possible dependency trees, which in the majority give spurious analyses. We show that, by using a POS tagger for choosing the most probable grammatical classes of the lexical units, we can substantially improve the rate of spurious ambiguity in a categorial dependency grammar of French developed by the NLP team of LINA. The experimental results show that our models perform better than the model which do not use a POS tagger at the cost of losing some correct analyses especially when the model of the tagger is very different to the lexical model of the parser.

1 INTRODUCTION

In the last years, dependency parsing becomes very popular and has been a topic of active research in natural language processing. Many different algorithms were suggested and evaluated for this task. They achieve both, a reasonable time complexity and a high accuracy. Statistical parsers with high accuracy are generally trained on texts annotated with morphological and sometimes also some other features. In particular, the minimal necessary annotation is POS tags. In this paper, we show how the use of POS tags may improve the rate of spurious ambiguity of parsing with a wide scope categorial dependency grammar of French (CDG) which uses L*efff* as its lexical base. In CDG, all lexical units (LU) are grouped into lexical classes (CDG classes). All units of a class share the same syntactic types. L*efff* is a wide coverage lexicon of French representing a very

large set of highly structured lexical information. Previously, a correspondence between CDG classes and Le*fff* classification was established and presented in [1].

The rest of the paper is structured as follows. Section 2 describes dependency grammars, Section 3 describes the parsing problem and our models. Section 4 presents the experimental evaluation, and Section 5 contains a comparative error analysis of the our different models. Finally, Section 6 concludes the paper.

2 DEPENDENCY GRAMMARS

Dependency-based representations have become increasingly popular in syntactic parsing, especially for languages that exhibit free or flexible word order, such as Czech (Collins et al., 1999), Bulgarian (Marinov Nivre, 2005), Turkish (Eryigit Oflazer, 2006), Russian (Boguslavsky et al., 2011). Many practical implementations of dependency parsing are restricted to projective structures, where the projection of a head word has to form a continuous substring of the sentence.

Dependency Grammars (DGs) are formal grammars assigning dependency trees (DT) to a sentence. A DT is a tree with words as nodes and dependencies, i.e. named syntactic binary relations between words, as arrows. In other words, if two words v_1 and v_2 are related by dependency d (denoted $v_1 \xrightarrow{d} v_2$) then v_1 is the governor and v_2 is the subordinate. Figure 1 illustrates the dependencies in the sentence "Au commencement était le Verbe."

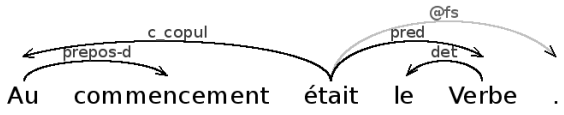


Fig. 1. *French: in the beginning was the Word.*

The relation $était \xrightarrow{pred} Verbe$ represents the predicative dependency between the copula *était* and the subject *Verbe*. The head of this sentence is *était*.

2.1 *Categorials Dependency Grammars*

Categorial Dependency Grammars introduced by [2] are lexicalized in the same sense as the conventional categorial grammars. Here we briefly give basic information on CDG.

The CDG types are defined over a set C of elementary categories (types). A syntactic type may be repetitive or optional $C^* =_{df} \{X^* | X \in C\}$, $C^? =_{df} \{X^? | X \in C\}$. CDG use iteration to express all kinds of repetitive dependencies such as modifiers and coordination relations.

The non-projective dependencies are expressed using *polarized valencies*. Namely, the governor G which has a right distant subordinate D through a discontinuous dependency d has positive dependency $\nearrow d$, whereas its subordinate D has the negative valency $\searrow d$. Together these dual valencies define the discontinuous dependency d.

In CDG, the anchor types of the form $\#(\searrow d)$, $\#(\swarrow d)$ are used in the same way as local dependencies. More precisely, CDG define discontinuous dependencies using polarized valencies (left / right, positive / negative) and a simple valencies pairing principle First Available (FA). For every valency, the corresponding one is the closest dual valency in the indicated direction.

In order to define polarized categories, we distinguish between four dependency polarities: left and right positive $\nwarrow, \nearrow$ and left and right negative $\searrow, \swarrow$. For each polarity $v \in \{\nwarrow, \searrow, \nearrow, \swarrow\}$ there is a unique dual polarity $\breve{v} : \breve{\nwarrow} = \swarrow, \breve{\swarrow} = \nwarrow, \breve{\nearrow} = \searrow, \breve{\searrow} = \nearrow$. $\nearrow C, \nwarrow C, \searrow C$ and $\swarrow C$ denote the corresponding sets of polarized distant dependency categories.

The general form of a CDG type is $[l_1 \setminus l_2 \setminus \cdots \setminus H / \cdots / r_2 / r_1]^P$ where the head type H defines the incoming dependency on the word, l_1 and r_1 are elementary (iterated or optional) categories which correspond to left or right outgoing dependencies or anchors, P is a potential, a string of polarized valencies which defines the long distance dependencies (incoming or outgoing), see [3], [4] and [5] for more details. Figure 2 shows two discontinuous dependencies (non-projective) in the sentence “elle la lui a donnée.”.

Categorial dependency grammars which define this dependency tree affect the types which anchor the clitics *la, lui* on the auxiliary *a*. The discontinuous dependencies are represented by dotted arrows.

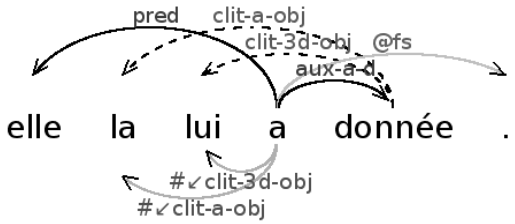


Fig. 2. Non-projective DS: "*she it[fem.] to him has given.".

$$
\begin{aligned}
&elle^{PN(Lex{=}pers,C{=}n)} && \mapsto [pred]\\
&la^{PN(Lex{=}pn,F{=}clit,C{=}a)} && \mapsto [\ \#(\swarrow\ clit{-}a\ -\ obj)]^{\swarrow clit{-}a{-}obj}\\
&lui^{PN(Lex{=}pn,F{=}clit,P{=}3,C{=}d)} && \mapsto [\ \#(\swarrow\ clit{-}3d\ -\ obj)]^{\swarrow clit{-}3d{-}obj}\\
&a^{Vaux(Lex{=}avoir,F{=}fin)} && \mapsto [\ \#(\swarrow\ clit{-}3d{-}obj)\\
& && \quad \backslash\#(\swarrow\ clit{-}a{-}obj)\\
& && \quad \backslash pred\backslash S/@fs/aux{-}a{-}d]\\
&donnée^{V2t(F{=}pz,C1{=}a,C2{=}d|g|l,T{=}past)} && \\
& && \mapsto [aux{-}a{-}d]^{\nwarrow clit{-}3d{-}obj \nwarrow clit{-}a{-}obj}\\
&.^{FullStop(Lex{=}".")} && \mapsto [@fs]
\end{aligned}
$$

The word *elle* is classified as a pronoun (PN), where *pers* and *n* correspond to person and noun. The word *la* is classified as a clitic at accusative case. The word *lui* is classified as a clitic for 3rd person with complement at dative case. The word *a* is classified as an auxiliary verb with a finite form "F=fin" while the word *donnée* is classified as a ditransitive verb where pz is "past participle" form and has two arguments (complement), the first complement is a direct complement (at accusative) and the second complement is a dative, a genitive or a locative.

The NLP team has developed a large scale CDG of French and a general purpose offline CDG parser. In this French CDG, the types are assigned to CDG classes (see [6] for details). The CDG parser is currently used to develop dependency tree corpora. The linguist's interface of this parser lets manually select for every LU one of its possible classes and one of the possible head dependencies. Then the parser finds all analyses compatible with the selection. Our goal in this paper is to automatically pre-fetch the most probable CDG classes per LU depending on its POS and to measure the impact of this selection on the ambiguity of the parser as applied to the CDG of French.

3 POS-BASED PARSING MODELS

Usually, the task of disambiguation of a dependency parser consists in deriving a single correct dependency tree τ for a given sentence S. The parsing problem consist in finding the mapping of an input sentence S, constituted of words $w_1 \cdots w_n$, to its dependency tree τ. More precisely, given a parsing model M and a sentence S, we derive the optimal dependency tree τ for S according to M. So the parsing problem is to construct the optimal dependencies for the input sentence, given the parsing model. Some parsers solve this problem by deriving a single analysis for each sentence. Our task is different: we should instead lower the ambiguity of the French CDG using POS tagging models and we evaluate the effect obtained by our method. Our POS-based parsing models first choose the most probable CDG classes through POS tags for the words in a sentence. Applying our method we should resolve a technical problem which arises from the nature of the lexical database of the CDG of French. In fact, this lexical database uses the (freely available) wide-coverage French lexicon Le*fff* [7]. It contains 110,477 lemmas (simple and compounds) and 536,375 inflected forms. The main part of the French CDG classes linked with Le*fff* is saved in a PostgreSQL Database. In this database, each LU of Le*fff* corresponds to one or several CDG classes. This correspondence is realized in the main table `lexicon`. Unfortunately, Le*fff* is not complete and contains errors. Therefore, in the lexical database there are several facilities for correction and complementation of Le*fff* definitions.

Before we describe our approach, we should explain that the CDG parser uses the following two strategies for lexicon (called below models):

Base model gives access to the forms contained in the classes of the French CDG (about 1500 forms), and also gives access to the original definitions of Le*fff* related with the CDG classes in the database.

The three other models use Le*fff* and the French CDG implicitly. First, a tagger is applied to the input sentence (Tree-Tagger [8] in T.T Model, MElt-Tagger [9] in M.T model and Brill tagger [10] in B.T model), Figure 4 presents this strategy.

Then, depending on the computed (composite in general) LU and their POS, a compatible lexical definition for every pair (LU, POS) and the corresponding CDG class is found in the database. If and when they exist, they are integrated to the input file that is sent to the parser.

Correspondence between POS tagging and Lefff: The correspondence between CDG classes and Le*fff* is established using the workspace dis-

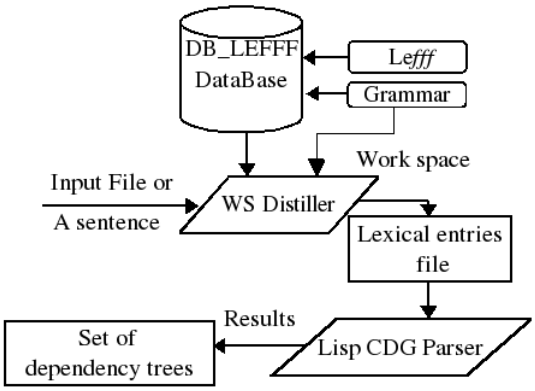


Fig. 3. General form of Base model.

tiller shown in Figure 4. We try to find the correspondence between the tags of POS-tagger and the syntactic categories of L*efff*. This correspondence is approximate, because the lexical models of POS-tagger and of L*efff* and the french CDG are different. Table 1 shows some examples of the correspondence.

Table 1. *Examples of correspondence between POS-tagger and* L*efff*.

L*efff*	T.T	M.T	B.T
np (noun phrases)	NAM	NPP	NAM, SBP
coo (coordination)	KON	CC, ET	COO
det (determiner)	DET:ART	DET	DTN
nc (commun nous)	NOM, NUM	NC	SBC, CAR

Some important information on POS-tagging e.g. `VER:futu` are very useful to determine both the *mood* and the *tense* of a verb. In this case, we also compare them to the *mood* and *tense* of the lexicon database. For instance, `VER:futu` means that *mood* is indicative and *tense* is future.

The WS distillers of the different models take an input file which contains the sentences with it POS (annotated sentence), and the output is a file with (lexical entries) annotated CDG classes and word features.

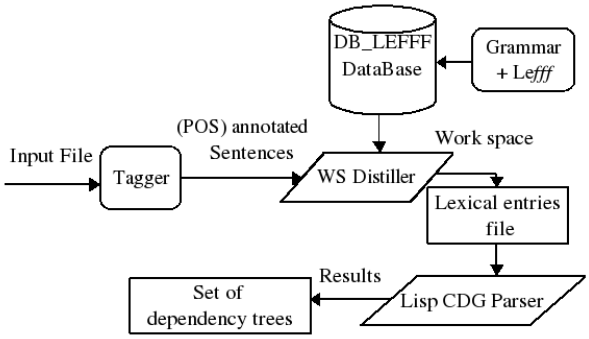


Fig. 4. General form of POS-bassed parsing models.

The algorithm chooses the most probable CDG classes for a LU by using POS tags and *cat* of Le*fff*.

This algorithm consists of the next three steps.

- First we search by the word of the sentence and its POS tag and compare them between the correspondence to *form* and it category that are found in the database, if it's equal, then we take it CDG class and it morphological features such as *mood*, *tense*, *person*, *gender*, *number*, *lemma* and saved all these information on file (lexical entry).
- If there is no result from the first step then we only search by the word of the sentence and compared it with *form* and take all the morphological information that correspond to this *form*.
- If also there is no result we classified this LU as "`UT(Lex=V|N|Adj|Adv)`". This CDG class is assigned to unknown LU.

4 Experimental Results

In our experiments we use a corpus of sentences divided into two subsets. The first subset, serving as a test set, consists of 1443 French sentences that have been analyzed to build the French Gold Standard dependency corpus (DTB): a corpus with French sentences from various sources. These sentences have 14974 projective and non projective (discontinuous) dependencies.

The second subset of the corpus has 184 French sentences from the French treebank [11].

For the experiment with the first subset, we first run the parser with the maximum number of viewed dependency trees set to 2000. We can not request all the possible dependency trees per sentence. With the French CDG, it generates hundreds of spurious structures per sentence. So for long and complex sentences, it is practically impossible to know how many DS are produced. Till the final step where the DS are extracted from the chart, the parsing algorithm is polynomial. Given that the number of these DS may be exponential with respect to the size of the chart, the final step is exponential in space in the worst case. In this step, the DS are generated from the chart in a certain order. The parser generates a HTML report page, which includes various useful statistics. It can also produce an XML structure representation of every DS including all necessary information.

For our POS-based parsing models, we compute the ambiguity reduction of dependency trees using the formula $X^j = \sum_{i=1}^{N} A_i^j$ where A_i^j is the number of dependency trees that are found for model j, where j is Base model, T.T model, M.T model or B.T model and i=1,...,N. N represents the number of the sentences that have a 100% correct analysis in every model. For our experiments, N=325. The reduction of dependency trees of model j is $\frac{X^{Base}-X^j}{X^{Base}} \times 100$, where j is different from the Base model. 2

Table 2. *Experimental results (dependency structures) compared to four models*

	Base	T.T	M.T	B.T
# DS for 325 sentences	153938	42572	44056	46718
Reduction # DS	% wrt model j	72.34%	71.38%	69.65%
geometric mean # DS$_j$/# DS$_{Base}$	-	0,24	0,23	0,26

We do not compute the number of dependency trees of the sentences that have more or eqaul 2000 analyses and also we just take into account the sentences that have at least one analysis for each parsing model. Table 3 shows some cases (accepted or canceled cases).

For the second experiment with the first subset, we run the parser with the maximum number of viewed dependency trees set to 1 in order to obtain the maximal number of analyzed sentences, and also to know how many sentences have all dependencies correctly analyzed. We compute

Table 3. *Cas canceled or accepted for the # DS.*

Cas canceled or accepted	Base	T.T	M.T	B.T
$\times$ (because 0 analysis)	55	34	55	0
$\times$ (because >=2000 analyse)	>=2000	666	1000	867
$\times$ (because no analyse)	11	9	no analyse	8
$\surd$ (accepted)	67	13	16	33

the total number of composition trees [1] using the formula $Y^j = \sum_{i=1}^{M} B_i^j$, where B_i^j is the number of composition trees for sentences that are found using model j, where j is Base model, T.T model, M.T model or B.T model and i=1,...,M. M represents the number of sentences that have at least one analysis in every model. For our experiments M=780. The reduction of the composition trees for model j is $\frac{Y^{Base}-Y^j}{Y^{Base}} \times 100$, where j is different from Base model.

Table 4. *Experimental results (composition trees) compared to four parsing models*

	Base	T.T	M.T	B.T
# CT for 780 sentences	16330×10^8	27×10^8	34×10^8	28×10^8
Reduction de # CT	% wrt model j	99.83%	99.79%	99.82%
geometric mean	-	0,035	0,037	0,033
# CT$_j$/# CT$_{Base}$				

The results in Tables 4 and 2 show that the numbers of composition trees and dependency trees of the three POS-based parsing models are inferior that of Base model. Our models achieve high reduction of both, composition trees and dependency trees (over 99% and 70% respectively).

The evaluation of the parser uses classical measures. It uses the labeled attachment score AS_L for the mode on Figure 4, which is the

[1] For each dependency tree, there are several composition trees because each composition tree specifies also a set of word features, a class and a type. We use the number of composition trees rather than the number of dependency trees, because it's usually not possible to evaluate the total number of dependency trees.

proportion of tokens that are assigned the correct head and the correct dependency label. The labeled attachment score represents the percentage of tokens that have been assigned both the correct head and the correct dependency label. There are several sentences which have accuracy over 90% of correct dependencies, but we count only the sentences that have 100% correct analysis. The result in Table 5 shows that our models achieve between 88% and 95% accuracy for correct dependency relation labeling.

Table 5. *Experimental results of parsing accuracy compared to four parsing models.*

	Base	T.T	M.T	B.T
# Sentences that have at least one analysis (1)	1089	1125	1005	949
# Sentences have 100% correct dependencies	1089	874	892	667
Recall	75.46%	60.65%	61.81%	46.22%
Precision	100%	77.68%	88.75%	70.28%
# of dependencies (from (1))	8255	9571	7730	7603
# correct dependencies	8255	8465	7380	6838
Recall correct dependencies	55.12%	56.53%	49.28%	45.66%
Precision	100%	88.44%	95.47%	89.93%
Labeled accuracy average (on all 1443 sentences)	100%	82.27%	85%	69%

We do not need to use unlabeled attachment score AS_U, because we don't compare the result of several parsers, AS_U is used by [12], that compare between two parsing architectures for the high accuracy on unknown words. Indeed BKY+FLABELER [13] achieves only a 82.56% tagging accuracy for the unknown words in the development set (5.96% of the tokens), whereas MElt+MST [14] achieves 90.01%.

Comparing between the three POS-based parsing models, we note that M.T model performs better than T.T and B.T models in terms of parsing accuracy. But T.T model is better than the other models in terms of ambiguity reduction and parsing time.

Table 6 shows an example to explain the reduction for both dependency trees and composition trees of the four parsing models in the sentence : "il parle en courtes phrases".

Table 6. *Reduction (dependency trees and composition trees) on the sentence "he speaks in short sentences".*

	Base	T.T	M.T	B.T
Reduction (# DS)	268	54	211	54
Reduction (# CT)	28732	3336	8559	3336

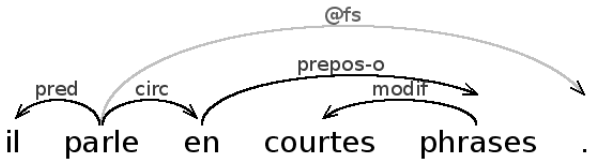


Fig. 5. "*he speaks in short sentences*"

$$
\begin{aligned}
il &\longmapsto PN(Lex = pers, C = n)\\
parle &\longmapsto Vt(F = fin, C = g)\\
en &\longmapsto PP(F = compl - obl, C = o)\\
courtes &\longmapsto Adj(F = modifier)\\
phrases &\longmapsto N(Lex = common)
\end{aligned}
$$

Table 7 shows comparative parsing times for each parsing model.

Table 7. *Comparation of the parsing times (four parsing models for 1443 sentences)*

	Base	T.T	M.T	B.T
Sentences that have at least one analysis	1089	1125	1005	949
Sentences that are analysed incorrect	0	141	127	314
Analyzed sentences total	1089	1266	1132	1263
Parsing time	03h 37mn	01h 32mn	02h 31mn	02h 8mn
Sentences that are not analyzed	354	177	311	180
Parsing time	05h 09mn	03h 35mn	05h 18mn	03h 00mn
Parsing time total	8h 46m	5h 07m	7h 49m	5h 08m

Table 8. *Effect of class pre-fetching (Paris 7 corpus).*

	Base	T.T
#CT	1097325498316350	7048627222816
#CT	10973254×10^8	70486×10^8
Total reduction #CT	% wrt Base model	99,9%
geometric mean of # $CT_{T.T}$/# CT_{Base}	-	0.002
# CT of the sentence Figure 6	17284	241
# DT of the sentence Figure 6	1295	68

For the second subset of 184 French sentences, we use only the Base model and T.T model. We only compute the number of composition trees using the same formula of the first subset. The results show that pre-fetching of CDG classes reduces the ambiguity with respect to composition trees more than 99%.

Table 8 summarizes the experimental results for Base model and T.T model for the number of composition trees.

The results given in Table 8 show that pre-fetching of classes reduces the ambiguity in terms of composition trees more than 99%.

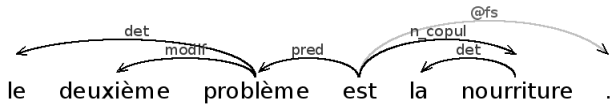


Fig. 6. *Paris 7 : the second problem is the food.*

5 DISCUSSION

This discussion provides a brief analysis of the errors made by the POS tagger for the first corpus, when we investigate the POS category of erroneous instances.

Each tagger tags this sentences by differnt way such as on the table 10.

In the CDG grammar, these tokens have different grammatical classes. As a result it gives different lexical classes for each token. Table 11 illustrates the lexical classes that correspond to the sentence “Ève, vas-t’en !”.

Table 9. *Errors make by the Parser for the parsing models.*

	memory exhansted	bad sentences	too complex sentences
Base	12	0	342
T.T	17	141	160
M.T	1	127	310
B.T	0	314	180

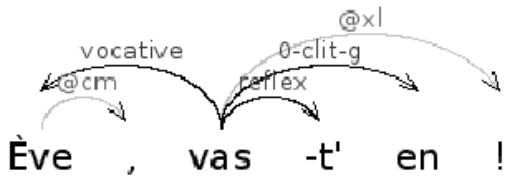


Fig. 7. *Structure de dépendances : Ève, vas-t'en !*

Table 10. Ťagset of differnt tagger

Tree Tagger	MElt Tagger	Bril Tagger
vas-t'en/NOM	vas-t'en/ADV	vas-t/VCJ
		en/PREP

Table 11. Classes assigned to the lexical unity

Frenche CDG	
lexical unity	Class
Éve	N(Lex=proper)
,	Comma(Lex='$CM')
vas	Vt(F=fin,C=l), Vt(F=fin,C=d), Vt(F=fin,C=a)
t'	PN(Lex=pn,F=refl)
en	PN(Lex=pn,F=clit,C=g—p), PN(Lex=attach-npers,C=g—p)
!	EmphatMark(Lex='!')

In the T.T model, there are 318 sentences that have no dependency tree, 177 sentences among them are not analyzed (time exceeded), which means there was not enough time to parse them, (the maximum number of seconds per sentence is set to 60 second), as we indicated above for ambiguous CDG. There are 141 sentences that are analysed as incorrect sentences. A first reason for this fact is that, there is at least one of

the next compound words in the sentences : *à peu près*, *Hé bien*, *dès lors*, *de loin*, *au dessous*, *la-bas*, *des EU*, *de l'*. In these cases, Tree-Tagger tags these compound words as separate words: *à* as *prep*, *peu* as *adv*, *près* as *adv*, etc. But the database has only complete entries for them. The second main reason is that Tree tagger makes errors in tagging for some LU. Thus the distiller do not find a good CDG class for these LU. We have seen that the results of B.T model are worse than those for T.T and M.T models, because Brill-Tagger also makes many errors in tagging. For example, the sentence "Adam ne donne à Ève pas que les pommes." (Adam do not give to Eve only apple) is annotated as `Adam/SBC:sg ne/ADV donne/SBC:sg à/PREP Ève/SBC:sg pas/ADV que/SUB les/DTN:pl pommes/SBC:pl ./.`. The verb *donne* is tagged as common noun ***SBC*** and not as a verb. There are 17 sentences contain *donne* tagged as ***SBC*** and 28 sentences which contain the past participle "été" of the verbe "être" are also tagged as ***SBC***. Errors like these lead to 314 sentences that have been analyzed as incorrect sentences. The example in Figure 6 shows the reason why we have obtained several analyses for this sentence. We note that the word "la", (*the*) is only tagged by T.T model as "determiner". Thus, there is only one CDG class corresponding to this LU: "`Det(Lex=art|pn)`", while Base model leaves all the CDG classes for this word. More precisely, the word *la* has in the grammar three different CDG classes, because this LU has different syntactic categories in L*efff* such as *det*, *nc* and *pro* as illustrated in Table 12.

Table 12. *Some features and classes in the Database for LU "La".*

Form	Cat	Class
la	cla	PN(Lex=pers,C=a)
la	cla	PN(Lex=pn,F=clit,C=a)
la	det	Det(Lex=art—pn)
la	nc	N(Lex=common)

This lexical ambiguity in Base model leads to several analyses of this sentence. This example shows the importance of the assignment of proper POS tag to every word in a sentence which is to be parsed.

In the one hand, the POS tagging reduces the search space for the parser, and also reduces ambiguity, improving parsing by limiting the search space. The sentences are also more often completely analyzed

by the parser, because the search space is smaller as compared to Base model.

On the other hand, using POS tagging, we lost some analyses for the reason of POS tagging errors. These sentences have been considered as incorrect sentences by the parser.

6 CONCLUSION

This paper evaluates the rate of improving dependency parsing through using different POS-tag models. These models choose the most probable grammatical classes for a word in a sentence based on POS tags, unfortunately at the cost of losing some correct analyses. Our experimental results have demonstrated the utility of POS-based parsing models. These models achieved substantial reductions of the number of dependency trees and of composition trees per sentence. Our experiments also show that to obtain an interesting system, the model used by the POS tagger must be compatible to the lexical model of the parser.

REFERENCES

1. Alfared, R., Béchet, D., Dikovsky, A.: "CDG LAB": a toolbox for dependency grammars and dependency treebanks development. In: K. Gerdes, E. Hajicova, and L. Wanner (Eds.), Proc. of the 1st Intern. Conf. on Dependency Linguistics (Depling 2011), Barcelona, Spain (September 2011)
2. Dikovsky, A.: Dependencies as categories. In: "Recent Advances in Dependency Grammars". COLING'04 Workshop. (2004) 90–97
3. Dekhtyar, M., Dikovsky, A.: Categorial dependency grammars. In Moortgat, M., ed.: Proceedings of Categorial Grammars 2004. (2004) 76–91
4. Béchet, D., Dikovsky, A., Foret, A.: Dependency structure grammar. In Philippe Blache, Edward Stabler, J.B., Moot, R., eds.: Proceedings of the 5th International Conference on Logical Aspects of Computational Linguistics, Bordeaux, France, April 2005. Volume 3492 of Lecture Notes in Artificial Intelligence (LNAI)., Springer-Verlag (April 2005) 18–34
5. Dekhtyar, M., Dikovsky, A.: Generalized categorial dependency grammars. In: Pillars of Computer Science'08. (2008) 230–255
6. Dikovsky, A.: Towards wide coverage categorical dependency grammars. In: Proceedings of the ESSLLI'2009 Workshop Parsing with Categorial Grammars - Parsing with Categorial Grammars Workshop ESSLLI 2009 Book of Abstracts. (2009) 230–255
7. Sagot, B.: The lefff, a freely available and large-coverage morphological and syntactic lexicon for french. In Calzolari (Conference Chair), N., Choukri,

K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., Tapias, D., eds.: Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta, European Language Resources Association (ELRA) (may 2010)
8. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Proceedings of International Conference on New Methods in Language Processing, Manchester, UK. (1994)
9. Denis, P., Sagot, B.: Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. In: Pacific Asia Conference on Language, Information and Computation, Hong Kong, Chine (2009)
10. Brill, E.: Some advances in transformation-based part of speech tagging. In: Proceedings of the twelfth national conference on artificial intelligence. (1994) 722–727
11. Abeillé, A., Barrier, N.: Enriching a french treebank. In: Proc. of LREC'04, Lisbon, Portugal (2004)
12. Candito, M., Crabbé, B., Denis, P.: Statistical french dependency parsing: Treebank conversion and first results. In Calzolari (Conference Chair), N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., Tapias, D., eds.: Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta, European Language Resources Association (ELRA) (may 2010)
13. Petrov, S., Barrett, L., Thibaux, R., Klein, D.: Learning accurate, compact, and interpretable tree annotation. In: ACL'06. (2006) 433–440
14. McDonald, R.: Discriminative Training and Spanning Tree Algorithms for Dependency Parsing. PhD thesis, University of Pennsylvania (July 2006)

RAMADAN ALFARED
LINA,
UNIVERSITY OF NANTES,
2, RUE DE LA HOUSSINIÈRE, 44000 NANTES, FRANCE
E-MAIL: <RAMADAN.ALFARED@ETU.UNIV-NANTES.FR>

DENIS BÉCHET
LINA,
UNIVERSITY OF NANTES,
2, RUE DE LA HOUSSINIÈRE, 44000 NANTES, FRANCE
E-MAIL: <DENIS.BECHET@UNIV-NANTES.FR>

IJCLA VOL. 3, NO. 2, JUL-DEC 2012, PP. 123–134
RECEIVED 31/10/11 ACCEPTED 09/12/11 FINAL 08/06/12

Exploring Self-training and Co-training for Dependency Parsing

RAHUL GOUTAM AND BHARAT RAM AMBATI

IIIT-H, India

ABSTRACT

We explore the effect of self-training and co-training on Hindi dependency parsing. We use Malt parser, which is a state-of-the-art Hindi dependency parser, and apply self-training using a large unannotated corpus. For co-training, we use MST parser with comparable accuracy to the Malt parser. Experiments are performed using two types of raw corpora—one from the same domain as the test data and another, which is out-of-domain from the test data. Through these experiments, we compare the impact of self-training and co-training on Hindi dependency parsing.

KEYWORDS**:** *Bootstrapping, dependency parsing, syntax, morphologically rich language.*

1 INTRODUCTION

Parsing morphologically rich free-word-order languages like Czech, Hindi, Turkish, etc., is a challenging task. Unlike English, most of the parsers for such languages have adopted the dependency grammatical framework. It is a well-known fact that for these languages, dependency framework is better suited [18, 12, 2]. Due to the availability of annotated corpora in recent years, data driven dependency parsing has achieved considerable success. In spite of availability of annotated treebanks, state-of-the art parsers for these

languages have not reached the performance obtained for English [14]. Frequently stated reasons for low performance are small treebank size, complex linguistic phenomena, long distance dependencies, and non-projective structures [14, 15, 3].

In this paper, we try to address the problem of small treebank size. We have lots of unannotated data. One way to increase treebank size is to manually annotate this data. But it is a very time consuming task. Another way is to parse this data using an existing parser and consider these automatic parses. But, what criteria should be used for extracting reliable parses from the automatically parsed data is a really challenging task.

In this paper, we explore the effect of two bootstrapping techniques, namely, self-training and co-training and see its impact on dependency parsing accuracy. We use Malt parser, that is the state-of-the-art Hindi dependency parser, and apply self-training using a large unannotated corpus. We also use MST parser with accuracy comparable to Malt parser and apply co-training.

We use two types of unannotated corpora, one from the same domain as the test data and another from a different domain, to explore the impact of domain of unannotated data on parsing accuracy. Though we work and present our results on Hindi, this approach can be applied to other languages with small treebanks like Telugu and Bangla.

This paper is arranged as follows. In Section 2, we describe the related work in bootstrapping in parsing. In Section 3, we present the state-of-the art Hindi dependency parser. In section 4, we report our experiments and analyze the results. We conclude with possible future work in Section 5.

2 RELATED WORK

In this section, we briefly describe the major works on bootstrapping in statistical dependency parsing.

The authors of [19] perform experiments to show that unannotated data can be used to improve the performance of statistical parsers by bootstrapping techniques. The focus of their paper is on co-training between two statistical parsers but they also perform self-training experiments with each of the two parsers. Although the results of self-training are not very encouraging, co-training experiments report modest improvement in parsing accuracy. They also perform cross-genre experiments to show that co-training is beneficial even when the seed data is from a different domain compared to the unannotated data.

The authors of [17] also perform self-training by using unannotated data from two different corpora - one in-domain and the other out-of-domain. They show that parser adaptability can be enhanced via self-training. They also report significant reduction in annotation cost and amount of work because a small manually annotated seed data is used.

The authors of [10] use a two phase parser-reranker system for self-training using readily available unannotated data. The two-phase parser reranker system consists of a generative parser and a discriminative reranker. They apply self-training on the generative parser only and not on the discriminative reranker and report significant improvement in accuracy over the previous state-of-the-art accuracy for Wall Street Journal parsing.

All the above mentioned works are on phrase structure parsing of English. There is an attempt at exploring usefulness of large raw corpus for dependency parsing by [5]. They could achieve considerable improvement over baseline for Chinese using only high confident edges instead of entire sentences. In our work the focus is dependency parsing of Hindi using a discriminative parser. We also explore how domain of data affects the parser performance.

3 HINDI DEPENDENCY PARSING

In ICON 2009 and 2010, two tools contests were held that focused on Indian Language dependency parsing [6, 7]. In these contests, rule-based, constraint based, statistical and hybrid approaches were explored towards building dependency parsers for Hindi. In 2009 contest, given the gold standard chunk heads, the task was to find dependencies between them. But in 2010 contest, given words with gold features like part-of-speech (POS) and morph information, the task was to find word level dependency parse. The ICON 2010 tools contest Hindi data consists of 2972, 543 and 321 sentences for training, development and testing with an average sentence length of 22.6. This data is a part of a larger treebank [4] which is under development. This is a news corpus taken from well-known Hindi news daily.

3.1 *Baseline (State-of-the-art) System*

We consider the best system [8] in ICON 2010 tools contest as the starting point. [8] used MaltParser [15] and achieved 94.5% Unlabeled

Attachment Score (UAS) and 88.6% Labeled Attachment Score (LAS). They could achieve this using liblinear learner and nivrestandard parsing algorithm. But, as mentioned above, POS and other features used in this system were gold standard. The only available system which uses automatically extracted features and does complete word level parsing for Hindi is [1]. Though both [1] and [8] used MaltParser, the data used is the subset of the one used by the latter and the parser settings were slightly different.

Table 1. Comparison of Different Systems

System	UAS	LAS	LS
Ambati et al. (2010)+ automatic features	85.5%	75.4%	78.9%
Kosaraju et al.(2010) + gold features	94.5%	88.6%	90.0%
Kosaraju et al.(2010) + automaticFeatures	86.5%	77.9%	81.7%

Taking training data and parser settings of Kosaraju et al. (2010) and automatic features similar to Ambati et al. (2010), we developed a parser and evaluated it on the ICON 2010 tools contest test data. We could achieve LAS of 77.9% and UAS of 86.5% on test set. This is the state-of-the-art system for Hindi dependency parsing using automatic features. We consider this system as our baseline and try to explore bootstrapping techniques to improve accuracy.

4 Experiments and Analysis

4.1 *Self-Training*

The parser used for self-training experiments is the Malt parser. We apply the settings of [8] along with automatic features (last line of Table 1). The parser is first trained on the ICON 2010 training data for Hindi. The model generated is then used to parse the unannotated corpus.

In the self-training experiments, we add the data incrementally in iterations. At each iteration, 1000 sentences are chosen randomly from the unannotated corpus which has been parsed by the model generated above and added to the training data. The parser is then trained again and the generated model is used to parse the test data.

Self-training experiments were performed using two types of data: one from the news domain (in-domain) and another from a different domain comprising mostly tourism data (out-of-domain).

4.1.1 *Self-Training: In-Domain*

We have taken unannotated news corpus of about 108,000 sentences. As a first step, we have cleaned the data. In this process, we removed the repeated sentences, and very large sentences (greater than 100 words per sentence).

Performance of the system on test data for the first 50 iterations is shown in Figures 1 and 2. Best accuracies of 78.6% LAS and 87% UAS were achieved, an improvement of 0.7% and 0.5% respectively.

4.1.2 *Self-Training: Out-of-Domain*

In this experiment, unannotated data from a domain different from the actual training and testing data is used for self- training. For this purpose, we have taken a non-news corpus of about 700,000 sentences. Similar to in-domain data, we first cleaned the data.

Performance of the resulting system on test data for the first 50 iterations is shown in Figures 1 and 2. There isnt any improvement in LAS over the baseline. Best accuracy of 77.8% LAS and 86.8% UAS was observed, an improvement of 0.3% in UAS, but a decrement of 0.1% in LAS.

4.2 *Co-Training*

The parsers used for co-training experiments are the Malt parser and the MST parser [6]. We have optimized the MST parser by modifying the feature extraction module so that the parser extracts relevant features for a morphologically rich language like Hindi. The best accuracy we achieved on the test set is 77.0% LAS and 86.5% UAS.

Using the best settings of the MST parser obtained above, a model is trained using the training set of ICON 2010 Hindi data with automatic features. This model is then used to parse the unannotated data. As in the self-training experiments, data are added incrementally in iterations. At each iteration, 1000 sentences are chosen randomly from the MST parser output and added to the training data of Malt parser. Malt parser is then trained again and the generated model is used to parse the test data.

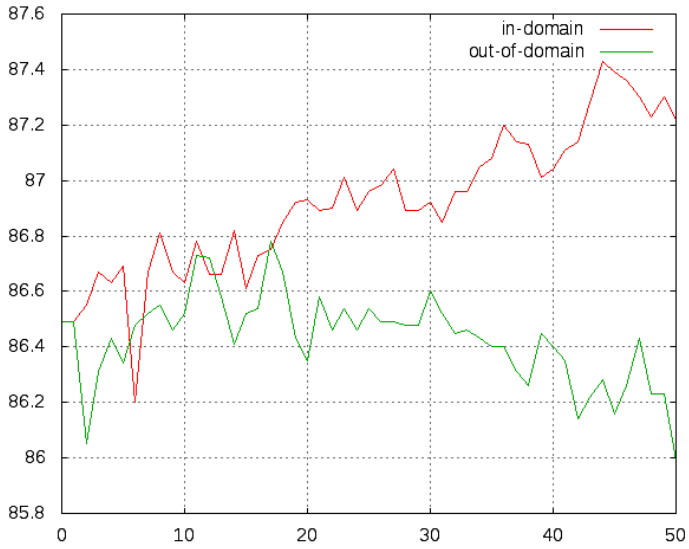


Fig. 1. Self-Training: UAS

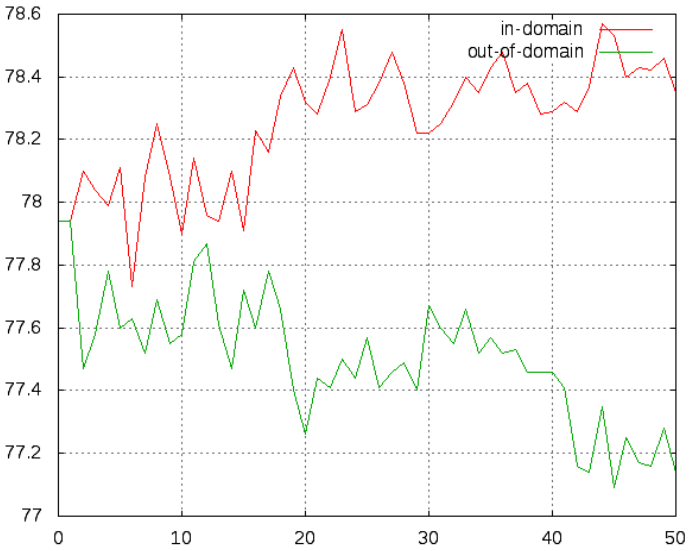


Fig. 2. Self-Training: LAS

4.2.1 *Co-Training: In-Domain*

The unannotated corpus used is the same as that used in self-training: in-domain experiments. Performance of the system is shown in Figures 3 and 4. Best accuracy of 78.6% LAS and 87.0% UAS was achieved, an improvement of 0.7% and 0.5%, respectively.

4.2.2 *Co-Training: Out-of-Domain*

The corpus used for out-of-domain experiments is the same as that used in self-training: out-of-domain experiments. Performance of the system is shown in Figures 3 and 4. There is a decrease in both UAS and LAS. The decrease in LAS is more compared to UAS.

4.3 *Co-Training: Sentence Selection via Agreement*

In this experiment, Malt and MST parsers are first trained using the training set of ICON 2010 Hindi data and then used to parse the unannotated data. The output of both parsers are then compared and sentences for which both Malt and MST parsers give the same parse are selected for bootstrapping. As in previous experiments, data is added incrementally with 1000 sentences per iteration. The 1000 sentences are chosen randomly from the pool of selected sentences and added to the training data of Malt parser. The parser is then trained again and the generated model is used to parse the test data.

4.3.1 *In-Domain Scenario*

The unannotated news corpus has approximately 108,000 sentences and both Malt and MST parsers gave the same parse for 10,461 sentences. These 10,461 sentences constitute our pool of selected sentences.

Performance of the system is shown in Figures 5 and 6. We achieved 78.8% LAS and 87.1% UAS, an improvement of 0.9% and 0.6% respectively over the baseline.

4.3.2 *Out-of-Domain Scenario*

The unannotated non-news corpus has approximately 700,000 sentences and both Malt and MST parsers gave the same parse for 45,328 sentences. These 45,328 sentences constitute our pool of selected sentences.

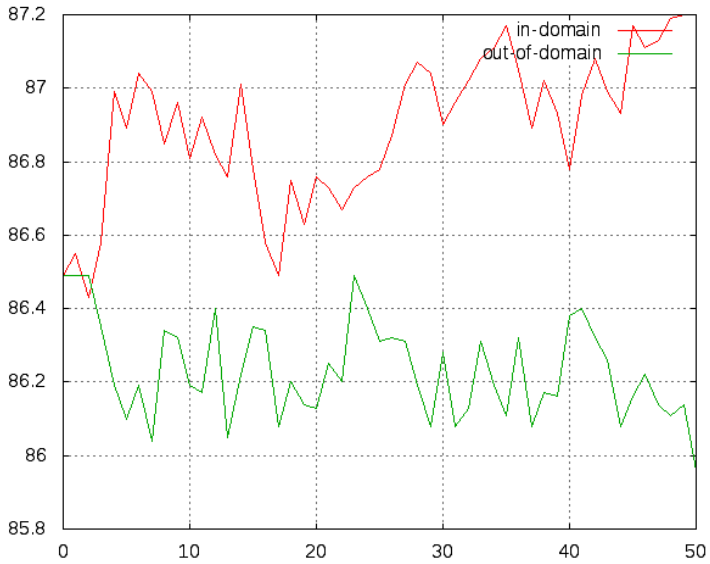


Fig. 3. Co-Training : UAS

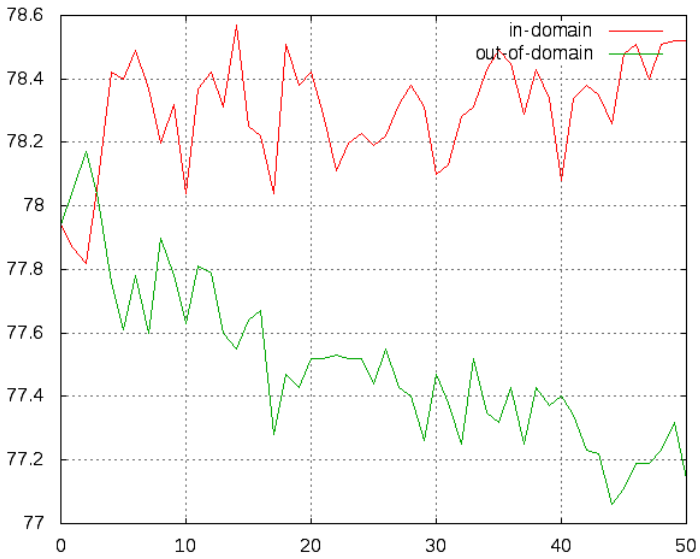


Fig. 4. Co-Training : LAS

Performance of the system for the first 12 iterations is shown in Figures 5 and 6. The remaining iterations are not shown because they follow a similar trend as the first few iterations. There is no improvement in LAS and UAS.

4.4. *Analysis*

Table 2 gives the summary comparing all the experiments performed. The * mark in the table shows that accuracy is statistically significant over the baseline. Significance is calculated using McNemar's test ($p \leq 0.05$) made available with MaltEval [13].

Table 2. Summary of experiments

System	UAS	LAS	LS
Baseline	86.5%	77.9%	81.7%
In-Domain Self-Training	87.0%*	78.6%*	82.3%*
Out-of-Domain Self-Training	86.8%	77.8%	81.6%
In-Domain Co-Training	87.0%*	78.6%*	82.2%*
Out-of-Domain Co-Training	86.5%	78.2%	82.0%
In-Domain Co-Training via Agreement	87.1%*	78.8%*	82.6%*
Out-of-Domain Co-Training v/ Agreement	86.5%	77.8%	81.6%

We could achieve significant improvement in accuracy over state-of-the-art system by applying bootstrapping with unannotated data from the same domain. There was a decrease in parser performance when data from a different domain was used. This clearly showed the importance of domain when applying bootstrapping in statistical parsers. Self-training and co-training both gave roughly the same improvement in performance for both UAS and LAS which is achieved after 23 iterations for self-training and 14 iterations for co-training. Co-training via agreement gave greater improvement in less number of iterations due to better sentence selection criteria.

We have also experimented with different sentence selection criteria. Classification scores were obtained for each labeled attachment for both the Malt and MST parsers. These scores represent the liblinear classification score for Malt and the maxent labeler probability for MST. These scores were then used to calculate the confidence score of a sentence.

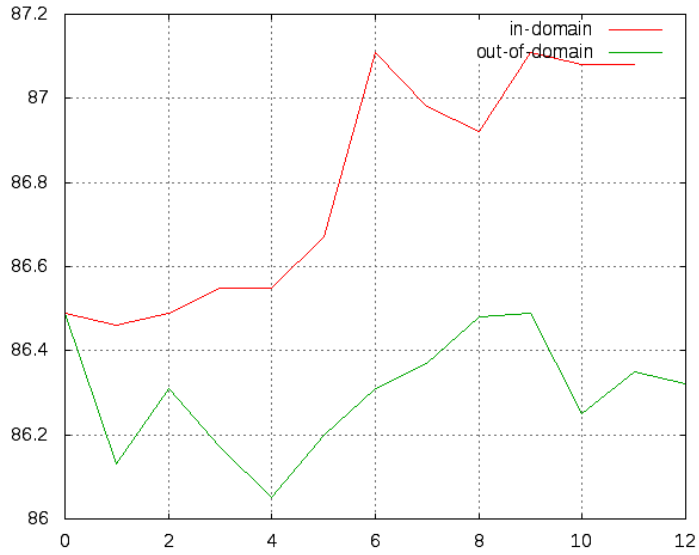


Fig. 5. Co-training v/ agreement: UAS

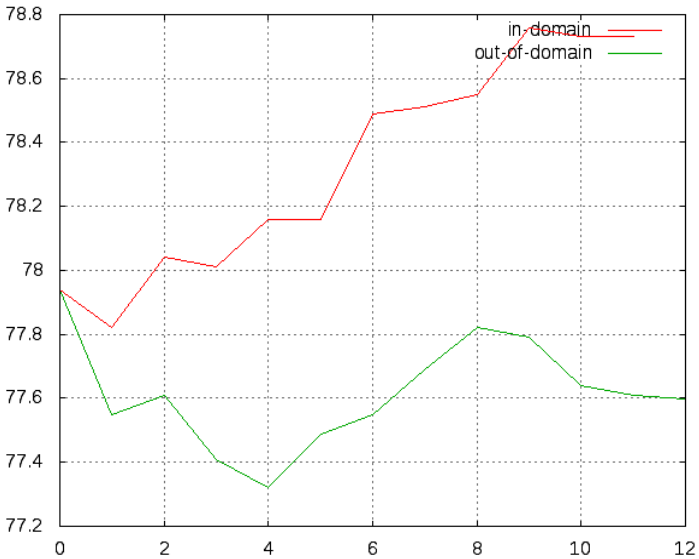


Fig. 6. Co-training v/ agreement: LAS

We have experimented with different methods to calculate confidence score of sentence, such as

- average score of labeled attachment,
- threshold on maximum and minimum score of all labeled attachments in sentence,
- normalized product,
- considering inter-chunk attachment scores only as accuracy of intra-chunk attachment is very high [1].

The most confident sentences were then added to the training data for the next iteration of bootstrapping. All these methods gave modest improvement, but the best improvement we could obtain was by selecting sentences via agreement between the two parsers.

We analyzed the label-wise precision of in-domain self-training experiments and found that there is significant improvement in precision of labels for which Malt parser is poor at identifying. For example, precision of label “main” (root of the sentence) increased from 65.4% to 84.8%. We observed two major reasons for it:

Increase in vocabulary. Approximately 30% of nodes correctly classified as “main” in the self-trained system (but not in the baseline system) are out-of-vocabulary words.

Most of the remaining cases were highly ambiguous that got correctly identified because of better feature tuning. In case of co-training, improvement in recall was observed across most labels, but there was a drop in precision.

5 Conclusion and Future Work

We explored the effect of applying bootstrapping techniques self-training and co-training on Hindi Dependency Parsing. We also performed in-domain and out-of-domain experiments to analyze the impact of domain on bootstrapping. We also explored different selection criteria and our results showed that the selection criteria need not be very sophisticated. Even random selection of sentences or simple agreement between the two parsers for sentence selection gives significant improvement in parsing accuracy.

In the future, instead of using whole sentence parse, we plan to use sub-parses that the parser is confident about to be used in

bootstrapping. We also plan to apply bootstrapping in other Indian languages such as Telugu and Bangla.

REFERENCES

1. Ambati, B. R., Gupta, M., Husain, S., Sharma, D. M.: 2010. A high recall error identification tool for Hindi treebank validation. Proceedings of The7th International Conference on Language Resources and Evaluation (LREC), Valleta, Malta.
2. Bharati, A., Chaitanya, V., Sangal, R.: 1995. Natural Language Processing: A Paninian Perspective. Prentice-Hall of India, New Delhi.
3. Bharati, A Husain, S., Ambati, B., Jain, S., Sharma, D., Sangal, R.: 2008. Two semantic features make all the difference in parsing accuracy. In Proceedings of ICON-08.
4. Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D. M., Xia, F. 2009.: Multi Representational and Multi-Layered Treebank for Hindi/Urdu. In proceedings of the Third Linguistic Annotation Workshop at 47 th ACL and 4th IJCNLP.
5. Chen, W., Wu, Y., Isahara, H.: 2008. Learning reliable information for dependency parsing adaptation. In Proceedings of the 22nd International Conference on Computational Linguistics (COLING).
6. Husain, S.: 2009. Dependency Parsers for Indian Languages. In Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing. Hyderabad, India.
7. Husain, S., Mannem, P., Ambati, B., Gadde, P.: 2010. The ICON- 2010 Tools Contest on Indian Language Dependency Parsing. In Proceedings of ICON-2010 Tools Contest on Indian Language Dependency Parsing.Kharagpur, India.
8. Kosaraju, P., Kesidi, S. R., Ainavolu, V. B. R., Kukkadapu, P.: 2010. Experiments on Indian Language Dependency Parsing. In Proc. of the ICON-2010 NLP Tools Contest: Indian Language Dependency Parsing.
9. Mannem, P., Dara, A.: 2011. Partial Parsing from Bitext Projections. In Proceedings of the 49th Annual Meeting of the Association of Computational Linguistics.
10. McClosky, D., Charniak, E., Johnson, M.: 2006. Effective Self- Training for Parsing. In Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics.
11. McDonald, R., Lerman, K., Pereira, F.: 2006. Multilingual dependency analysis with a two-stage discriminative parser. In Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X), pp. 216220.
12. Mel'čuk, I. A.: 1988. Dependency Syntax: Theory and Practice. State University Press of New York.

13. Nilsson, I. J., Nivre, J. 2008. Malteval: An evaluation and visualization tool for dependency parsing. In Proceedings of the Sixth LREC, Marrakech, Morocco.
14. Nivre, J., Hall, J., Kubler, S., McDonald, R., Nilsson, J., Riedel, S., Yuret, D. 2007a. TheCoNLL 2007 Shared Task on Dependency Parsing. In Proceedings of EMNLP/CoNLL-2007.
15. Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kbler, S., Marinov, S., Marsi, E. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. Natural Language Engineering, 13(2), 95-135.
16. Nivre, J., Rimell, L., McDonald, R., GmezRodrguez, C. 2010. Evaluation of Dependency Parsers on Unbounded Dependencies.In proceedings of the International Conference on Computational Linguistics (COLING).
17. Reichart, R., Rappoport, A. 2007. Self-Training for Enhancement and Domain Adaptation of Statistical Parsers Trained on Small Datasets. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics.
18. Shieber, S. M. 1985. Evidence against the contextfreeness of natural language. In Linguistics and Philosophy, p. 8, 334343.
19. Steedman, M. Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlen, P., Baker, S., Crim, J. 2003. Bootstrapping Statistical Parsers from Small Datasets.In Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics Volume 1.
20. Aho, A. V., Ullman, J. D. 1972. The Theory of Parsing, Translation and Compiling, volume 1. Prentice-Hall, Englewood Cliffs, NJ.
21. American Psychological Association. 1983. Publications Manual. American Psychological Association, Washington, DC.
22. Association for Computing Machinery. 1983. Computing Reviews, 24(11):503–512.
23. Chandra, A. K., Kozen, D. C., Stockmeyer, L. J. 1981. Alternation. Journal of the Association for Computing Machinery, 28(1):114–133.
24. Gusfield, D. 1997. Algorithms on Strings, Trees and Sequences. Cambridge University Press, Cambridge, UK.

RAHUL GOUTAM
LANGUAGE TECHNOLOGIES RESEARCH INSTITUTE, IIIT-H,
OBH-208, GACHIBOWLI, HYDERABAD, 500032, INDIA.
E-MAIL: <RAHUL.GOUTAM@RESEARCH.IIIT.AC.IN>

BHARAT RAM AMBATI
LANGUAGE TECHNOLOGIES RESEARCH INSTITUTE, IIIT-H,
OBH-208, GACHIBOWLI, HYDERABAD, 500032, INDIA.
E-MAIL: <AMBATI@RESEARCH.IIIT.AC.IN>

IJCLA VOL. 3, NO. 2, JUL-DEC 2012, PP. 135–150
RECEIVED 30/11/12 ACCEPTED 10/12/12 FINAL 12/12/12

Entity Linking by Leveraging Extensive Corpus and Semantic Knowledge

YUHANG GUO, BING QIN, TING LIU, AND SHENG LI

Harbin Institute of Technology, China

ABSTRACT

Linking entities in free text to the referent knowledge base entries, namely, entity linking is attractive because it connects unstructured data with structured knowledge. An essential part of this task is the modeling of the entity. Several methods have been proposed to this problem, but they suffer from the sparseness problem. In this paper, we present a new approach to the entity modeling. This approach models an entity by leveraging extensive entity-related corpus to overcome the sparseness, and alleviates the data imbalance between popular and unpopular entities by smoothing. Furthermore, we propose a novel model for the entity linking, which combines contextual relatedness and semantic knowledge. Experimental results on two benchmark data sets show that our proposed approach outperforms the state-of-the-art methods significantly.

KEYWORDS: *Entity Linking, Data Imbalance, Smoothing, Semantic Knowledge*

1 INTRODUCTION

Bridging unstructured text with structured knowledge is widely needed in many natural language processing and data mining tasks. In recent years, as large scale knowledge bases (e.g. Wikipedia[1]) become available, the

[1] http://www.wikipedia.org

entity linking task, which links named entities in free text to the referent knowledge base entries, is attracting more and more attentions.

The major challenge of entity linking is that in natural language a name may refer to different entities in different contexts (i.e. the name ambiguity). In the past, many disambiguation methods have been proposed and gained certain success[1–7]. An essential part of the task is entity modeling. In previous methods, an entity is usually modeled as bag-of-words to measure the contextual similarity between the entity and the surrounding text. In the bag-of-words model, the entity is represented in a term vector of the corresponding entity-description text (e.g. the content of the entity's Wikipedia page). The term here may indicate a word, a named entity or a phrase. However, due to the limited amount of the entity-description text, such model suffers from sparseness problem. Therefore, additional features are incorporated to enhance this model, such as Wikipedia category tags[1], topics[8–12] and neighboring entities[2, 13–18]. However, these features depend on specific knowledge bases[1], need high complexity computation[13] and also suffer from the sparseness problem.

On the other hand, a virtue of the modern knowledge bases (e.g. Wikipedia, DBpedia[19], etc.) is that they contain not only large amount of entities but also massive internal links. In Wikipedia, the number of internal links is over 25 times as the number of articles[2]. These links directly lead a reader to the pages of the entities which are mentioned in the article. Assuming that an entity is related to the text where it is linked, all such texts can be harvested and combined as the training text of this entity. Here we call the combined entity-related training text *entity document*.

However, the above method brings a new problem: the data imbalance. Because popular entities are usually linked by more articles than unpopular ones, the *entity document* size of the popular entities is much bigger than the unpopular entities. This highly skewed distribution will harm system performance. In previous, the training data has to be reduced due to the data imbalance although potentially useful information may be lost [2, 7].

In this paper, we propose an approach to alleviate the data imbalance problem without the data reduction. Our approach is based on language model smoothing. Specifically, we compare two smoothing methods: Jelinek-Mercer smoothing[20] and Dirichlet prior smoothing[21].

[2] http://stats.wikimedia.org/EN/TablesWikipediaEN.htm

The two methods perform similarly in traditional information retrieval [22]. Interestingly, in entity linking the Dirichlet prior smoothing is efficient to the data imbalance problem and outperforms the Jelinek-Mercer smoothing.

Moreover, since the objective of entity linking is to find the referent entity rather than the most contextually related texts, the effect of semantic features should not be underestimated. However, in the basic language model, little semantic information is used. In this paper, we propose a novel probabilistic model which we call *alias model*. This model can not only capture contextual relatedness between the context and entity but also leverage semantic knowledge in the context to distinguish the referent entity from other contextually related entities. Evaluation on two benchmark data sets indicates that our proposed method performs better than the state-of-the-art entity linking significantly.

2 Problem and Approach Overview

Let $\mathcal{E}$ be the set of all entities in the real world. $\mathcal{K} \subseteq \mathcal{E}$ is a knowledge base. Each entity $e \in \mathcal{K}$ has a set of attributes, such as names/aliases, description texts and cross references to other entities, etc.

The entity linking problem is the following: for a name mention m in a given query document d, find the referent entity e in $\mathcal{K}$, if $e \notin \mathcal{K}$ return `NIL`. The query name mention and the query document constitute a query as the input and the referent entity or `NIL` is the expected output.

We evaluate our approach on two data sets: KBP2009 and KBP2010, which are taken from the Knowledge Base Population (KBP) Track[23, 4, 24]. KBP2009 contains 3,904 queries, in which all the query documents are newswire articles. KBP2010 contains 2,250 queries. The query documents of 1,500 queries are newswire articles and the rest 750 are web texts. KBP2009 and KBP2010 share the same knowledge base which is derived from Wikipedia and contains 818,741 entities. In both of the data sets, over a half of the referent entities (2229/3904 and 1230/2250) are absent from the track knowledge base and should be labeled as `NIL`.

The entity linking task can be broken down into two steps: candidate generation and candidate ranking. For the first step, the system selects candidate entities which may be represented in the form of the query name. This step reduces the cost from computing all entities down to a much smaller set of entities. For the second step, the system ranks the candidates and output the top rank candidate. In this paper, we mainly focus on the ranking method and use a simple method to detect the `NIL`

answer: if the candidate set is empty or the top ranked entity is absent from the track knowledge base then return `NIL`.

3 CANDIDATE GENERATION

The goal of the candidate generation is to obtain as many potential entities as possible for the given query name. Wikipedia provides *disambiguation pages* and *redirect pages* from which the candidates can be obtained. However the coverage of this method is not enough for this task because the query name may not be included in the *disambiguation pages* and the *redirect pages*. In this work, we explore the name variations for each entity in Wikipedia and construct a name-entity mapping. The candidates are then generated from this mapping directly.

Given an entity, we extract its names from the following sources in Wikipedia: *title*, *redirect page titles*, *disambiguation page titles*, *bold text* in the first paragraph of the entity, *name field* in the Infobox (e.g. *"name"*, *"birth_name"* or *"nick_name"*), and the *anchor text* of the hyperlinks which link to the entity.

We use the Jun. 20, 2011 version of English Wikipedia dump. In all 140.7 million name-entity pairs which contain 17.3 million names and 3.7 million entities. The name-entity mapping also includes the co-occurrence frequency of the name-entity pairs in Wikipedia. We published this data so that researchers can reproduce our results.

In general, acronym name is more ambiguous than the relevant full name and hence is more difficult to be disambiguated. For example, the acronym *ABC* can be mapped to 79 entities in our name-entity mapping. Meanwhile, the full name *All Basotho Convention* is unambiguous. Fortunately, in some cases the acronyms can be extended to their full forms according to the query document. The following cases are considered:

- The acronym is in a pair of parentheses and the full name is in front of the acronym. (e.g. ... *the newly-formed All Basotho Convention (ABC) is far from certain* ...)
- The full name is in a pair of parentheses and the acronym is in front of the full name. (e.g. ... *at a time when the CCP (Chinese Communist Party) claims* ...)
- The acronym consists of the initial letters of the full name words. (e.g. ... *leaders of Merkel's Christian Democratic Union* ... *CDU* ...)

Given a query name, if it is an acronym, we first attempt to extend its acronym in the query document and then substitute the query name with

the full name. We search the query name in the name-entity mapping and obtain the corresponding candidate entities.

The candidate generation recalls for the non-NIL queries are 91.6% and 94.9% on KBP2009 and KBP2010 data set respectively. Table 1 shows the number of all unique candidates and the average number of candidates per query.

Table 1. Result of the candidate generation on KBP2009 and KBP2010 data set.

Data Set	KBP2009	KBP2010
# of queries	3904	2250
# of non-NIL queries	1675	1020
# of unique candidates	7706	23682
# of candidates/query	22	35
Recall of non-NIL queries	91.6%	95.4%

4 CANDIDATE RANKING

In this section, we present a probabilistic model for the candidate ranking. Next we show the data imbalance between popular and unpopular entities and present how to alleviate the imbalance in the model estimation. Then we propose a novel probabilistic model for entity linking, the *alias model*, which can improve system performance by leveraging semantic knowledge.

4.1 *Probabilistic Model*

In this model, a document is considered as "generated" from a word distribution (e.g. language model). In this sense, the query document d is generated in the following steps: the document author first chooses the entity in mind (the knowledge base), as well as the corresponding name he/she wants to present, and then selects contextual words according to the language model of the entity.

Formally, let e and m denote the referent entity and the mention to be disambiguated. The objective function of entity linking is:

$$e^* = \arg\max_e P(e, m)P(d|e) \tag{1}$$

where $P(e, m)$ is the prior probability of e and m, and $P(d|e)$ is the generative probability of d from the model of e.

Let $f(e, m)$ denote the co-occurrence frequency of entity e and its mention m in the name-entity mapping. The maximum likelihood estimation of the prior probability is

$$P(e, m) = \frac{f(m, e)}{\sum_{e', n'} f(n', e')} \tag{2}$$

where $e' \in \mathcal{K}$, $n' \in N(e')$. $N(e')$ is the set of all names of e'. $P(e, m)$ is the probability of a random observed entity-name pair (e', n') is just the pair (e, m) we concern.

The unigram language model assigns the probability

$$P(d|e) = \prod_{t \in d} P(t|\theta_e) \tag{3}$$

This is the likelihood of the query document d according to the model of entity e. $P(t|\theta_e)$ is the probability of document term t generated by the model of e. Here we assume the terms are sampled from a multinomial distribution. The model parameter θ_e is the multinomial distribution parameter over terms.

The query document is modeled as a bag of terms surrounding the mention m within a window in d. In our approach a term is a name in the name-entity mapping. In our experiment, we extract the terms from the document by using forward maximum matching algorithm which is adopted from word segmentation [25]. We set the window size 50 according to the setting of [26].

Let $C(e)$ denote a bag of terms taken from the training text of e. Let $c(t, C(e))$ be the count of t in $C(e)$. The maximum likelihood estimation of $P(t|\theta_e)$ is

$$P_{ML}(t|\theta_e) = \frac{c(t, C(e))}{|C(e)|} \tag{4}$$

where $|C(e)| = \sum_{t' \in V} c(t', C(e))$ is the length of the training text and V is the set of all names in the name-entity mapping.

4.2 *Data Imbalance*

In previous methods, the entity model is usually trained by using the entity-description text (e.g. the Wikipedia page of the entity). However,

this modeling suffers from sparseness because the lengths of many entity-description texts are not enough to train robust models. In this work, we overcome the sparseness by leveraging extensive corpus (i.e. the *entity document*). The corpus is automatically derived from all the articles which contain a hyperlink leading to the entity to be modeled. However, the *entity document* lengths vary dramatically from popular entities to unpopular entities (e.g. from millions of terms to several terms). Figure 1 shows the *entity document* length distribution on two data sets. Note that the vertical axis is in log scale. From the curves we can see that the distribution approximately obeys Zipf's law[27]. On the both data sets, the longest *entity document* is `United States`, which contains 118 million terms. The average *entity document* length on KBP2009 and KBP2010 are 1.61×10^5 and 1.42×10^5, respectively. And the standard deviations are 1.70×10^6 and 1.20×10^6, respectively. This means that the *entity document* length distribution is highly imbalanced.

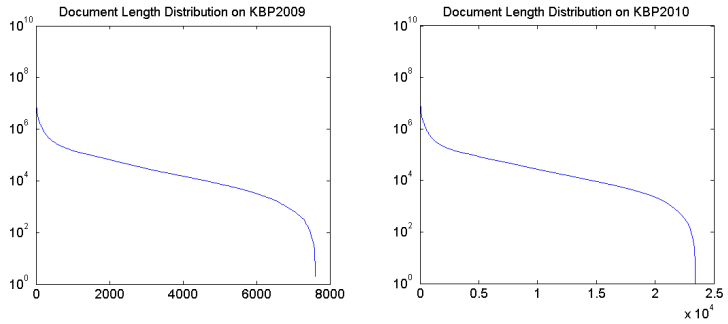


Fig. 1. *Entity document* length distribution over all the candidates on KBP2009 and KBP2010. The horizontal axis is the candidates ranked by the *entity document* length. The vertical axis is the *entity document* length in logarithm scale.

The data imbalance problem is common in knowledge bases because popular entries always have longer description texts and are cited by more articles than unpopular ones. As the growth of the knowledge bases, this information gap will be even larger. Because highly skewed distribution will harm system performance, the training corpus has to be reduced: drop some of the citation articles[2], or set a window around the citation of the entity[7]. Obviously, in this way some useful information about the entity will be lost. In this paper, we propose to employ all the entity-

related text for the training and alleviate the imbalance based on smoothing method.

4.3 *Smoothing*

Because in the maximum likelihood estimation $P_{ML}(t|\theta_e)$, the probability of unseen terms in $C(e)$ is zero, assigning non-zero probability according to some "background knowledge" to the unseen terms (i.e. smoothing) is critical to the accuracy of the model. The "background knowledge" here is the occurrence probability of t in the whole training corpus collection, which is called background model.

$$P(t|\theta_b) = \frac{\sum_{e' \in \mathcal{K}} c(t, C(e'))}{\sum_{e' \in \mathcal{K}} |C(e')|} \tag{5}$$

where θ_b denotes the parameter of the background model.

A direct smoothing is to combine the maximum likelihood estimate and the background model by linear interpolation. This method is also called Jelinek-Mercer smoothing (JM)[20], which is widely used in traditional information retrieval.

$$P(t|\theta_e) = \lambda P_{ML}(t|\theta_e) + (1 - \lambda) P(t|\theta_b) \tag{6}$$

where $\lambda \in (0, 1)$ is a smoothing parameter to control the proportion of the background model.

JM assigns the same background model proportion for each entity. However, if a small λ is assigned to a "long" (*entity document* length) entity, the distribution feature of the entity will be diluted by the background model. On the other hand, if a big λ is assigned to a "short" entity, the estimate of the entity model will be sparse. Since the *entity document* length varies dramatically, it is difficult to find a proper λ to provide good estimates for both "long" and "short" entities.

An alternate smoothing, Dirichlet prior smoothing (DP)[21], adds background model into the conjugate prior of the language model. The smoothed estimate is

$$P(t|\theta_e) = \frac{|C(e)|}{|C(e)| + \mu} P_{ML}(t|\theta_e) + \frac{\mu}{|C(e)| + \mu} P(t|\theta_b) \tag{7}$$

where μ is a smoothing parameter.

Comparing with JM, DP also interpolates maximum likelihood estimate with background model. But the interpolation coefficient in DP is

affected by the length of the *entity document*. For a fixed μ, the model of a "short" entity will be close to the background model, and the model of a "long" entity will be close to the maximum likelihood estimate. Therefore, both of the "short" and the "long" entities can benefit from this estimation at the same time. In the experiment section, we will show that the DP can alleviate the data imbalance and outperforms JM significantly.

4.4 *Alias Model*

Language modeling approaches can capture contextual relatedness between texts. However, a drawback of basic language model is that only the term features are used. In this work, we combine the contextual language model and more discriminative semantic features in a probabilistic framework.

Intuitively, if several different names/aliases or an unambiguous name of a candidate is observed in the document, the confidence on this candidate will increase. How to incorporate the name features into the model is a problem. To this end, we propose a alias model which highlights the name variations of the referent entity in the document.

Let n denote one of the names/aliases of e, we have

$$P(e,m,d) = \sum_{n \in N(e)} P(n,e,m,d) = P(e,m)P(d|e) \sum_{n \in N(e)} P(n|e,d) \tag{8}$$

where $P(e,m)$ and $P(d|e)$ can be estimated as in the basic language model. We approximate the sum factor by

$$\sum_{n \in N(e)} P(n|e,d) = \sum_{n \in N(e) \bigcap d} P(n|e) \tag{9}$$

The maximum likelihood estimation of $P(n|e)$ is

$$P(n|e) = \frac{f(n,e)}{f(e)} = \frac{f(n,e)}{\sum_{n' \in N(e)} f(n',e)} \tag{10}$$

where $f(e)$ is the frequency of entity e in the name-entity mapping.

Table 2 shows an example of how the basic language model can be improved by the name variations in the query document. In this example, the query name mention is *UT* and the referent entity is `University of Tampa`. In the basic language model, the score[3] of `University`

[3] Here the score is $\log(P(e,m,d))$.

of Texas at Austin is much higher than other candidates including University of Tampa because the former entity is more contextual related to the query document. But if we notice that an alias, *Tampa*, of University of Tampa appears in the query document, our confidence on linking *UT* to University of Tampa will be higher. In the alias model the score of University of Tampa increases and is higher than University of Texas at Austin. The name variations such as *Tampa* may appear in the language model of the string University of Tampa too, but the weight of these important features will be diluted by the large size of the *entity document*. In the alias model, such semantic discriminative terms are emphasized separately.

Table 2. An example of the model comparison (using Dirichlet prior smoothing).

Candidates (e)	$f(n,e)$		$f(e)$	$\sum P(n\|e)$	BLM score	AM score
	UT	*Tampa*				
University of Texas at Austin	14	0	6,901	0.0020	–256.44	–262.64
University of Tampa	3	114	449	0.2606	–260.66	–262.00

5 EXPERIMENTS

The evaluation metric is micro-averaged accuracy across the query set, that is, the proportion of the queries which are labeled the correct entity id in knowledge base (or NIL) by the system.

In order to compare with the previous systems, the first evaluation was conducted on KBP2009. We compared our methods with the top three system performances in the track (Siel_093, QUANTA1 and htlcoe1) and four systems reported in [7]:

- The cosine similarity-based method on bag of words features: BoW [2];
- The link similarity based method: TopicIndex[28];
- The improved link based method using machine learning techniques to balance the semantic relatedness, commonness and context quality: Learning2Link[29];

– The entity-mention model proposed by [7]: EMM. EMM is a language model based system with Jelinek-Mercer smoothing but is trained on balanced training data.

The systems that we implemented are trained on extensive training texts, including the basic language model based methods using Jelinek-Mercer smoothing and Dirichlet prior smoothing respectively: BLM-JM, BLM-DP and the alias model using the two smoothing methods: AM-JM, AM-DP.

The system performances on KBP data sets are shown in Table 3, where the three columns represent the system performance on: All queries (All), in-knowledge-base-answer queries (inKB) and NIL-answer queries (NIL) respectively. The results of BLM-JM, AM-JM, BLM-DP and AM-DP are optimal over the smoothing parameters which we have searched. AM-DP* on KBP2009 is the empirical optimal AM-DP result tuned on KBP2010 and AM-DP* on KBP2010 is tuned on KBP2009. The optimal values of λ and μ are shown in Table 4.

Table 3. Results on the KBP data set.

(a) KBP2009

	All	inKB	NIL
Siel_093	0.82	0.77	0.86
QUANTA1	0.80	0.77	0.83
hltcoe1	0.79	0.71	0.87
BoW	0.72	0.77	0.65
TopicIndex	0.80	0.65	0.91
Learning2Link	0.83	0.73	0.90
EMM	0.86	0.79	0.90
BLM-JM	0.84	0.75	0.91
AM-JM	0.86	0.77	0.92
BLM-DP	0.87	**0.81**	0.91
AM-DP	**0.88**	**0.81**	**0.93**
AM-DP*	**0.88**	**0.81**	**0.93**

(b) KBP2010

	All	inKB	NIL
LCC	0.86	0.79	**0.91**
Siel	0.82	0.72	0.90
CMCRC	0.82	0.74	0.89
KL	0.85	0.81	0.87
BLM-JM	0.84	0.79	0.89
AM-JM	0.85	0.79	0.90
BLM-DP	**0.88**	**0.84**	0.90
AM-DP	**0.88**	**0.84**	**0.91**
AM-DP*	**0.88**	**0.84**	**0.91**

As can be seen from Table 3(a), on KBP2009, our proposed method (i.e. AM-DP*) outperforms the best ranking system in the KBP track 2009 by 6% improvement. Compared with the BoW, TopicIndex, and Learning2Link baselines, our proposed method gets 16%, 8%, 5% im-

provements respectively. Our method also performs significantly better than the state-of-the-art method: EMM (under Z-test with $p < 0.01$).

Table 4. Optimal parameters of Jelinek-Mercer (JM) smoothing and Dirichlet prior (DP) smoothing on KBP2009 and KBP2010 data sets.

Smoothing	JM(λ)		DP($\mu \times 10^6$)	
Data Set	2009	2010	2009	2010
BLM	0.8	0.9	3.7	4.0
AM	0.75	0.7	3.7	4.0

Table 3(b) shows the system performances on KBP2010. We compared our method with the top three systems in the track (LCC, Siel and CMCRC) and a recently proposed KL-divergence based method: KL[30]. Our proposed method also outperforms the best system significantly ($p < 0.05$).

On the both data sets, the alias model performs better than the basic language model. DP outperforms JM significantly ($p < 0.01$) in the basic language model (up to 4%). AM-DP outperforms BLM-JM significantly by 4% ($p < 0.01$.).

The improvement of the alias model on KBP2009 is more than that on KBP2010. This is because the KBP2009 query documents are all news articles which are rich in names whereas the KBP2010 query documents consist of news and web text. Therefore, the number of effective aliases in each document of KBP2009 are more than KBP2010 on average. This indicates that the alias model performs better on the query documents which are rich in names.

6 CONCLUSIONS

In this paper, we propose to use extensive entity-related corpus to overcome the sparseness problem in the entity modeling of entity linking. Due to the highly skewed distribution of the entities, the training data for the entity modeling is highly imbalanced. We investigate language modeling based approaches and find that Dirichlet prior smoothing performs better than Jelinek-Mercer smoothing because it can leverage the length of entity entity's training text. The property of Dirichlet prior smoothing makes it suitable for the data imbalance scenario. We further combine the

contextual language modeling and name variation feature in a probabilistic framework and propose an alias model. Experimental results on two standard test sets show that the Dirichlet prior smoothing performs better than Jelinek-Mercer smoothing and our proposed model outperforms the state-of-the-art performance significantly.

ACKNOWLEDGEMENTS This work was supported by National Natural Science Foundation of China (NSFC) via grant 61273321, 61073126, 61133012 and the National 863 Leading Technology Research Project via grant 2012AA011102.

REFERENCES

1. Bunescu, R.C., Pasca, M.: Using encyclopedic knowledge for named entity disambiguation. In: EACL, The Association for Computer Linguistics (2006)
2. Cucerzan, S.: Large-scale named entity disambiguation based on Wikipedia data. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Prague, Czech Republic, Association for Computational Linguistics (June 2007) 708–716
3. Mihalcea, R., Csomai, A.: Wikify!: linking documents to encyclopedic knowledge. In: CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, New York, NY, USA, ACM (2007) 233–242
4. Dredze, M., McNamee, P., Rao, D., Gerber, A., Finin, T.: Entity disambiguation for knowledge base population. In: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), Beijing, China, Coling 2010 Organizing Committee (August 2010) 277–285
5. Zheng, Z., Li, F., Huang, M., Zhu, X.: Learning to link entities with knowledge base. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, California, Association for Computational Linguistics (June 2010) 483–491
6. Zhang, W., Sim, Y.C., Su, J., Tan, C.L.: Entity linking with effective acronym expansion, instance selection, and topic modeling. In Walsh, T., ed.: IJCAI 2011, Chiang Mai, Thailand (November 2011) 1909–1914
7. Han, X., Sun, L.: A generative entity-mention model for linking entities with knowledge base. In: 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies, Portland, Oregon, USA, Association for Computational Linguistics (June 2011) 945–954
8. Cucerzan, S.: TAC entity linking by performing full-document entity extraction and disambiguation. In: Text Analysis Conference 2011. (2011)

9. Kataria, S.S., Kumar, K.S., Rastogi, R.R., Sen, P., Sengamedu, S.H.: Entity disambiguation with hierarchical topic models. In: 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '11, New York, NY, USA, ACM (2011) 1037–1045
10. Zhang, W., Su, J., Tan, C.L.: A Wikipedia-LDA model for entity linking with batch size changing instance selection. In Walsh, T., ed.: IJCAI 2011, Chiang Mai, Thailand (November 2011) 562–570
11. Sen, P.: Collective context-aware topic models for entity disambiguation. In: 21st international conference on World Wide Web, WWW '12, New York, NY, USA (2012) 729–738
12. Han, X., Sun, L.: An entity-topic model for entity linking. In: 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Korea, Association for Computational Linguistics (July 2012) 105–115
13. Kulkarni, S., Singh, A., Ramakrishnan, G., Chakrabarti, S.: Collective annotation of Wikipedia entities in web text. In: 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '09, New York, NY, USA, ACM (2009) 457–466
14. Han, X., Sun, L., Zhao, J.: Collective entity linking in web text: a graph-based method. In: 34th international ACM SIGIR conference on Research and development in Information, SIGIR '11, New York, NY, USA (2011) 765–774
15. Hoffart, J., Yosef, M.A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., Weikum, G.: Robust disambiguation of named entities in text. In: 2011 Conference on Empirical Methods in Natural Language Processing, Edinburgh, Scotland, UK. (July 2011) 782–792
16. He, J., de Rijke, M., Sevenster, M., van Ommering, R., Qian, Y.: Generating links to background knowledge: a case study using narrative radiology reports. In: 20th ACM international conference on Information and knowledge management, CIKM '11, New York, NY, USA (2011) 1867–1876
17. Ratinov, L., Roth, D., Downey, D., Anderson, M.: Local and global algorithms for disambiguation to Wikipedia. In: 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, Oregon, USA (June 2011) 1375–1384
18. Shen, W., Wang, J., Luo, P., Wang, M.: Linden: linking named entities with knowledge base via semantic knowledge. In: 21st international conference on World Wide Web, WWW '12, New York, NY, USA (2012) 449–458
19. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: a nucleus for a web of open data. In: 6th international conference The semantic web and 2nd Asian conference on Asian semantic web, ISWC'07/ASWC'07, Berlin, Heidelberg, Springer-Verlag (2007) 722–735
20. Jelinek, F., Mercer, R.L.: Interpolated estimation of Markov source parameters from sparse data. In: Workshop on Pattern Recognition in Practice, Amsterdam, The Netherlands, North-Holland (May 1980) 381–397

21. MacKay, D.J.C., Peto, L.C.B.: A hierarchical dirichlet language model. Natural Language Engineering **1** (1995) 289–308
22. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. ACM Trans. Inf. Syst. **22** (April 2004) 179–214
23. McNamee, P., Dang, H.: Overview of the tac 2009 knowledge base population track. `http://www.nist.gov/tac/publications/2009/presentations/TAC2009_KBP_overview.pdf` (2009)
24. Ji, H., Grishman, R.: Knowledge base population: Successful approaches and challenges. In: 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, Oregon, USA, Association for Computational Linguistics (June 2011) 1148–1158
25. Guo, J.: Critical tokenization and its properties. Comput. Linguist. **23** (December 1997) 569–596
26. Pedersen, T., Purandare, A., Kulkarni, A.: Name discrimination by clustering similar contexts. In Gelbukh, A., ed.: CICLing 2005, Lecture Notes in Computer Science. Volume 3406., Springer (2005) 226–237
27. Zipf, G.K.: Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology. Addison-Wesley (1949)
28. Medelyan, O., Witten, I.H., Milne, D.: Topic indexing with Wikipedia. In: AAAI WikiAI workshop. Volume 1., AAAI Press (2008) 19–24
29. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management, New York, NY, USA, ACM (2008) 509–518
30. Gottipati, S., Jiang, J.: Linking entities to a knowledge base with query expansion. In: 2011 Conference on Empirical Methods in Natural Language Processing, Edinburgh, Scotland, UK., Association for Computational Linguistics (July 2011) 804–813

YUHANG GUO
RESEARCH CENTER FOR SOCIAL COMPUTING AND INFORMATION RETRIEVAL,
HARBIN INSTITUTE OF TECHNOLOGY,
CHINA
E-MAIL: <YHGUO@IR.HIT.EDU.CN>

BING QIN
RESEARCH CENTER FOR SOCIAL COMPUTING AND INFORMATION RETRIEVAL,
HARBIN INSTITUTE OF TECHNOLOGY,
CHINA
E-MAIL: <BQIN@IR.HIT.EDU.CN>

TING LIU
(CORRESPONDING AUTHOR)
RESEARCH CENTER FOR SOCIAL COMPUTING AND INFORMATION RETRIEVAL,
HARBIN INSTITUTE OF TECHNOLOGY,
CHINA
E-MAIL: <TLIU@IR.HIT.EDU.CN>

SHENG LI
RESEARCH CENTER FOR SOCIAL COMPUTING AND INFORMATION RETRIEVAL,
HARBIN INSTITUTE OF TECHNOLOGY,
CHINA
E-MAIL: <SLI@IR.HIT.EDU.CN>

Applications

IJCLA VOL. 3, NO. 2, JUL-DEC 2012, PP. 153–166
RECEIVED 03/11/11 ACCEPTED 09/12/11 FINAL 08/06/12

Improving Finite-State Spell-Checker Suggestions with Part of Speech N-Grams

TOMMI A. PIRINEN, MIIKKA SILFVERBERG,
AND KRISTER LINDÉN

University of Helsinki, Finland

ABSTRACT

We demonstrate a finite-state implementation of context-aware spell checking utilizing an N-gram based part of speech (POS) tagger to rerank the suggestions from a simple edit-distance based spell-checker. We demonstrate the benefits of context-aware spell-checking for English and Finnish and introduce modifications that are necessary to make traditional N-gram models work for morphologically more complex languages, such as Finnish.

1 INTRODUCTION

Spell-checking by computer is perhaps one of the oldest and most researched applications in the field of language technology starting from the mid 20th century [1]. One of the crucial parts of spell-checking—both from an interactive user-interface point of view and for unsupervised correction of errors—is the production of spelling suggestions. In this article we test various finite-state methods for using context and shallow morphological analysis to improve the suggestions generated by traditional edit distance measures or unigram frequencies such as simple weighted finite-state dictionaries trained from word form frequencies as in [2].

The spell-checking task can be split into two parts, i.e. *detection* and actual *correction* of the spelling errors. The spelling errors can be detected in text as word forms that are unlikely to belong to the natural language in question, such as writing 'cta' instead of 'cat'. This form of spelling errors is commonly called *non-word (spelling) errors*. Another

form of spelling errors is word forms that do not belong to the given context under certain syntactic or semantic requirements, such as writing 'their' instead of 'there'. This form is correspondingly called *real-word (spelling) errors*. The non-word type of spelling errors can easily be detected using a dictionary, whereas the detection of the latter type of errors typically requires syntactic analysis or probabilistic methods [3]. For the purpose of this article we do not distinguish between them, as the same correction methods can be applied to both.

The correction of spelling errors usually means generating a list of word forms belonging to the language for a user to chose from. The mechanism for generating correction suggestions for the erroneous word-forms is an *error-model*. The purpose of an error-model is to act as a filter to revert the mistakes the user typing the erroneous word-form has made. The simplest and most traditional model for making such corrections is the Levenshtein-Damerau edit distance algorithm, attributed initially to [4] and especially in the context of spell-checking to [1]. The Levenshtein-Damerau edit distance assumes that spelling errors are one of insertion, deletion or changing of a single character to another, or swapping two adjacent characters, which models well the spelling errors caused by an accidental slip of finger on a keyboard. It was originally discovered that for most languages and spelling errors, this simple method already covers 80 % of all spelling errors [1]. This model is also language-independent, ignoring the differences in character repertoires of a given language. Various other error models have also been developed, ranging from confusion sets to phonemic folding [5].

In this paper, we evaluate the use of context for further fine-tuning of the correction suggestions. The context is still not commonly used in spell-checkers. According to [5] it was lacking in the majority of spell-checkers and while the situation may have improved slightly for some commercial office suite products, the main spell-checkers for open source environments are still primarily context-ignorant, such as hunspell[1] which is widely used in the open source world. For English, the surface word-form trigrams model has been demonstrated to be reasonably efficient both for non-word cases [6] and for for real-word cases[7]. As an additional way to improve the set of suggestions, we propose to use morphosyntactically relevant analyses in context. In this article, we evaluate a model with a statistical morphological tagger [8]. The resulting system is in effect similar as described in [9] for Spanish[2].

[1] `http://hunspell.sf.net`

The system described is fully built on freely available tools and data, available for download and use[3]. The only exception to this is the training data for Finnish, since there is no available morphological training data for Finnish as of yet, the download does not contain the source material for training but only the trigram models compiled into binary format automata.

Furthermore, we test the context-based spelling methods using both English and Finnish language materials to ensure the applicability of the method for morphologically different languages. The reason for doing this is two-fold; firstly the fact that English has rather low morphological productivity may make it behave statistically differently from other languages. On the other hand, English has the largest amount of freely available text corpora. For other languages, the availability of free corpora, especially annotated material, is often seen as a problem.

The article is laid out as follows: In Section 2, we outline the implementation of a finite-state context-aware spell-checker and describe the statistical methods used. In Section 3, we introduce the corpora and dictionaries used for spell-checking and training material as well as the corpora used for obtaining the spelling errors with context. In Section 4, we show how the created spelling correctors improve the results and explain the errors left. In Section 5, we compare our work with other current systems and enumerate possible improvements for both.

2 METHODS

The spelling correction in this article is performed in several phases: assuming misspelled word *cta* for *cat*, we first apply the error model to the already known incorrect string *cta* to produce candidates for probable mistypings. For this purpose we use the Damerau-Levenshtein edit-distance algorithm in finite-state form. When applied to *cta* we get all strings with one or two typing mistakes, i.e. *ata*, *bta*, ..., *acta*, *bcta*, ..., *ta*, *ca*, ..., *tca*, and the correct *cat*. This set of strings is simultaneously matched against the language model, which will produce a set of corrections, such as *cat*, *act* or *car*. Since both the error-model and the language model contain information on likelihoods of errors and words

[2] We are grateful for the anonymous reviewer on bringing this previous work on same methods and similar systems to our knowledge.

[3] From the page `http://hfst.svn.sourceforge.net/viewvc/hfst/trunk/cicling-2011-contextspell/`

respectively, the resulting list will be sorted according to a combination of the edit distance measure and the probability of the word in a reference corpus. The rankings based on edit distance alone and the edit distance combined with word probabilities form our two baseline models.

The context-based models we introduce here use the suggestion list gained from a contextless spelling-checker and the context of the words as input to rerank suggestions based on N-gram models. Each of the suggestions is tried against the N-gram models, and the ones with higher likelihoods will be lifted. For example when correcting the misspelling of 'an' as 'anx' in the sentence "this is anx example sentence", as shown in the Table 1, we have the surface trigrams {this, is, _}, {is, _, example}, {_, example, sentence}, and corresponding analysis trigrams {DET, VVBZ, _}, {VVBZ, _, NN}, {_, NN, NN}. The suggestions for anx at edit distance one include 'ax', 'an' (one deletion), 'ant', 'and', 'any' (one change) and so on. To rank the possible suggestions, we substitute s_3 with the suggestions, and calculate the likelihood of their analyses.

Table 1. Example trigram combinations

this$_{s_1}$	is$_{s_2}$	_$_{s_3}$	example$_{s_4}$	sentence$_{s_5}$
DET$_{a_1}$	VVBZ$_{a_2}$	_ a_3	NN$_{a_4}$	NN$_{a_5}$

2.1 *Weighted Finite-State Interpretation of the Method*

In this article we use a finite-state formulation of spell-checking. We assume the standard notation for finite-state algebra and define the language model as a weighted finite-state automaton assigning a weight to each correctly spelled word-form of a language, and an error model automaton mapping a misspelled string to a set of corrected strings and their weights. The probabilistic interpretation of the components is such that the weighted fsa as a language model assigns weight $w(s)$ to word s corresponding to the probability $P(s)$ for the word to be a correct word in the language. The error model assigns weight $w(s:r)$ to string pair s, r corresponding to the probability $P(s|r)$ of a user writing word r when intending to write the word s, and the context model assigns weight $w(s_3a_3)$ to word s_3 with associated POS tagging a_3 corresponding to the standard HMM estimate $P(a_3s_3)$ of the analysis being in a 3-gram

context given by equation (1).

$$P(a_3 s_3) = \prod_{i=3}^{5} P(s_i|a_i) P(a_i|a_{i-2},\ a_{i-1}) \tag{1}$$

In a weighted finite-state system, the probabilistic data needs to be converted to the algebra supported by the finite-state weight structure. In this case we use the tropical semi-ring by transforming the frequencies into penalty weights with the formula $-\log \frac{f}{CS}$, where f is the frequency and CS the corpus size in number of tokens. If the language model allows for words that are not in the dictionary, a maximal weight is assigned to the unseen word forms that may be in the language model but not in the training corpus, i.e. any unseen word has a penalty weight of $-\log \frac{1}{CS}$.

The spelling corrections suggested by these unigram lexicon-based spell-checkers are initially generated by composing an edit-distance automaton [10] with an error weight corresponding to the probability of the error estimated in a corpus, i.e. $-\log \frac{f_F}{CS+1}$, where f_F is the frequency of the misspelling in a corpus. This weight is attached to the edit distance type error. In practice, this typically still means that the corrections are initially ordered primarily by the edit distance of the correction, and secondarily by the unigram frequency of the word-form in the reference corpus. This order is implicitly encoded in the weighted paths of the resulting automaton; to list the corrections we use the n-best paths algorithm [11]. This ordering is also used as our second baseline.

For a context-based reordering of the corrections, we use the POS tagging probabilities for the given suggestions. The implementation of the analysis N-gram probability estimation is similar to the one described in [8] with the following adaptations for the spelling correction. For the suggestion which gives the highest ranking, the most likely analysis is selected. The N-gram probability is estimated separately for each spelling suggestion and then combined with the baseline probability given by the unigram probability and the edit distance weight. The ideal scaling for the weights of unigram probabilities, i.e. edit distance probabilities with respect to N-gram probabilities, can be acquired by e.g.g tuning the scaling parameter on an automatically generated spelling error corpus.

The resulting finite-state system consists of three sets of automata, i.e. the dictionary for spell-checking, the error-model as described in [2], and the new N-gram model automata. The automata sizes are given in Table 2 for reference. The sizes also give an estimate of the memory usage of the spell-checking system, although the actual memory-usage

during correction will rise depending on the actual extent of the search space during the correction phase.

Table 2. Automata sizes.

Automaton	States	Transitions	Bytes
English			
Dictionary	25,330	42,448	1.2 MiB
Error model	1,303	492,232	5.9 MiB
N-gram lexicon	363,053	1,253,315	42 MiB
N-gram sequences	46,517	200,168	4.2 MiB
Finnish			
Dictionary	179,035	395,032	16 MiB
Error model	1,863	983,227	12 MiB
N-gram lexicon	70,665	263,298	8.0 MiB
N-gram sequences	3,325	22,418	430 KiB

2.2 *English-Specific Finite-State Weighting Methods*

The language model for English was created as described in [12][4]. It consists of the word-forms and their probabilities in the corpora. The edit distance is composed of the standard English alphabet with an estimated error likelihood of 1 in 1000 words. Similarly for the English N-gram material, the initial analyses found in the WSJ corpus were used in the finite-state tagger as such. The scaling factor between the dictionary probability model and the edit distance model was acquired by estimating the optimal multipliers using the automatic misspellings and corrections of a Project Gutenberg Ebook[5] *Alice's Adventures in Wonderland*. In here the estimation simply means trying out factors until results are stable and picking the best one.

2.3 *Finnish-Specific Finite-State Weighting Methods*

The Finnish language model was based on a readily-available morphological weighted analyser of Finnish language [13]. We further modified

[4] The finite-state formulation of this is informally described on the following page: `http://blogs.helsinki.fi/tapirine/2011/07/21/how-to-write-an-hfst-spelling-corrector/`

[5] `http://www.gutenberg.org/cache/epub/11/pg11.txt`

the automaton to penalize suggestions with newly created compounds and derivations by adding a weight greater than the maximum to such suggestions, i.e. $-A \log \frac{1}{CS+1}$, where A is the scaling factor acquired from the training material. This has nearly the same effect as using a separate dictionary for suggestions that excludes the heavily weighted forms without requiring the extra space. Also for Finnish, a scaling factor was estimated by using automatic misspellings and corrections of a Project Gutenberg Ebook[6] *Juha*.

In the initial Finnish tagger, there was a relatively large tagset, all of which did not contain information necessary for the task of spell-checking, such as discourse particles, which are relatively context-agnostic [14], so we opted to simplify the tagging in these cases. Furthermore, the tagger used for training produced heuristic readings for unrecognized word-forms, which we also removed. Finally, we needed to add some extra penalties to the word forms unknown to the dictionary in the N-gram model, since this phenomenon was more frequent and diverse for Finnish than English. The extra penalties were acquired by iterative testing on the correction material using generated errors.

3 MATERIAL

To train the spell-checker lexicons, word-form probabilities can be acquired from arbitrary running text. By using unigram frequencies, we can assign all word-forms some initial probabilities in isolation, i.e. with no spell-checking context. The unigram-trained models we used were acquired from existing spell-checker systems [12, 2], but we briefly describe the used corpora here as well.

To train the various N-gram models, corpora are required. For the surface-form training material, it is sufficient to capture running N-grams in the text. For training the statistical tagger with annotations, we also require disambiguated readings. Ideally, this means hand-annotated tree banks or similar gold standard corpora.

The corpora used are summarized in Table 3. The sizes are provided to make it possible to reconstruct the systems. In practice, they are the newest available versions of the respective corpora at the time of testing. In the table, the first row is the training material used for the finite-state lexicon, i.e. the extracted surface word-forms without the analyses for

[6] See the page `http://www.gutenberg.org/cache/epub/10863/pg10863.txt`

unigram training. The second row is for the analyzed and disambiguated material for the N-gram based taggers for suggestion improvement. The third line is the corpora of spelling errors used only for the evaluation of the systems. As we can see from the figures of English compared with Finnish, there is a significant difference in freely available corpora such as Wikipedia. When going further to lesser resourced languages, the number will drop enough to make such statistical approaches less useful, e.g. Northern Sámi in [2].

Table 3. Sizes of training and evaluation corpora.

	Sentences	Tokens	Word-forms
English			
Unigrams		2,110,728,338	128,457
N-grams	42,164	969,905	39,690
Errors	85	606	217
Finnish			
Unigrams		17,479,297	968,996
N-grams	98,699	1,027,514	144,658
Errors	333	4,177	2,762

3.1 *English corpora*

The English dictionary is based on a frequency weighted word-form list of the English language as proposed in [12]. The word-forms were collected from the English Wiktionary[7], the English EBooks from the project Gutenberg[8] and the British National Corpus[9]. This frequency weighted word-list is in effect used as a unigram lexicon for spell-checking.

To train an English morphosyntactic tagger, we use the WSJ corpus. In this corpus each word is annotated by a single tag that encodes some morphosyntactic information, such as part-of-speech and inflectional form. The total number of tags in this corpus is 77.

The spelling errors of English were acquired by extracting the ones with context from the Birkbeck error corpus[10]. In this corpus, the errors are from a variety of sources, including errors made by children

[7] `http://en.wiktionary.org`
[8] `http://www.gutenberg.org/browse/languages/en`
[9] `http://www.kilgarriff.co.uk/bnc-readme.html`
[10] `http://ota.oucs.ox.ac.uk/headers/0643.xml`

and language-learners. For the purpose of this experiment we picked the subset of errors which had context and also removed the cases of word joining and splitting to simplify the implementation of parsing and suggestion. When interpreting results it should be noted that many of these English errors are competence errors while the baseline algorithm used to model errors here is for typing errors.

3.2 *Finnish Corpora*

As the Finnish dictionary, we selected the freely available open source finite-state implementation of a Finnish morphological analyser[11]. The analyser had the frequency-weighted word-form list based on Finnish Wikipedia[12] making it in practice an extended unigram lexicon for the Finnish language. The Finnish morphological analyser, however, is capable of infinite compounding and derivation, which makes it a notably different approach to spell checking than the English finite word-form list.

The Finnish morphosyntactic N-gram model was trained using a morphosyntactically analyzed Finnish Newspaper[13]. In this format, the annotation is based on a sequence of tags, encoding part of speech and inflectional form. The total number of different tag sequences for this annotation is 747.

For Finnish spelling errors, we ran the Finnish unigram spell-checker through Wikipedia, europarl and a corpus of Finnish EBooks from the project Gutenberg[14] to acquire the non-word spelling errors, and picked at random the errors having frequencies in range 1 to 8 instances; a majority of higher frequency non-words were actually proper nouns or neologisms missing from the dictionary. Using all of Wikipedia, europarl and Gutenberg provides a reasonable variety of both contemporary and old texts in a wide range of styles.

4 Tests and Evaluation

The evaluation of the correction suggestion quality is described in Table 4. The Table 4 contains the precision for the spelling errors. The precision is measured by ranked suggestions. In the tables, we give the results

[11] `http://home.gna.org/omorfi`
[12] `http://download.wikipedia.org/fiwiki/`
[13] `http://www.csc.fi/english/research/software/ftc`
[14] `http://www.gutenberg.org/browse/languages/fi`

separately for ranks 1—5, and then for the accumulated ranks 1—10. The rows of the table represent different combinations of the N-gram models. The first row is a baseline score achieved by the weighted edit distance model alone, and the second is with unigram-weighted dictionary over edit-distance 2. The last two columns are the traditional word-form N-gram model and our POS tagger based extension to it.

Table 4. Precision of suggestion algorithms with real spelling errors.

Algorithm	1	2	3	4	5	1—10
English						
Edit distance 2 (baseline)	25.9 %	2.4 %	2.4 %	1.2 %	3.5 %	94.1 %
Edit distance 2 w/ Unigrams	28.2 %	5.9 %	29.4 %	3.5%	28.2 %	97.6 %
Edit distance 2 w/ Word N-grams	29.4 %	10.6 %	34.1 %	5.9 %	14.1 %	97.7 %
Edit distance 2 w/ POS N-grams	68.2 %	18.8 %	3.5 %	2.4 %	0.0 %	92.9 %
Finnish						
Edit distance 2 (baseline)	66.5 %	8.7 %	4.0 %	4.7 %	1.9 %	89.8 %
Edit distance 2 w/ Unigrams	61.2 %	13.4 %	1.6 %	3.1 %	3.4 %	88.2 %
Edit distance 2 w/ Word N-grams	65.0 %	14.4 %	3.8 %	3.1 %	2.2 %	90.6 %
Edit distance 2 w/ POS N-grams	71.4 %	9.3 %	1.2 %	3.4 %	0.3 %	85.7 %

It would appear that POS N-grams will in both cases give a significant boost to the results, whereas the word-form N-grams will merely give a slight increase to the results. In the next subsections we further dissect the specific changes to results the different approaches give.

4.1 *English Error-Analysis*

In [12], the authors identify errors that are not solved using simple unigram weights, such as correcting *rember* to *remember* instead of *member*. Here, our scaled POS N-gram context-model as well as the simpler word N-gram model, which can bypass the edit distance model weight will select the correct suggestion. However, when correcting e.g. *ment* to *meant* in stead of *went* or *met* the POS based context reranking gives no help as the POS stays the same.

4.2 *Finnish Error-Analysis*

In Finnish results we can easily notice that variation within the first position in the baseline results and reference system is more sporadic. This

can be traced back to the fuzz factor caused by a majority of probabilities falling into the same category in our tests. The same edit-distance and unigram probability leaves the decision to random factors irrelevant to this experiment, such as alphabetical ordering that comes from data structures backing up the program code. The N-gram based reorderings are the only methods that can tiebreak the results here.

An obvious improvement for Finnish with POS N-grams comes from correcting agreeing NP's towards case agreement, such as *yhdistetstä* to *yhdisteistä* ('of compounds' PL ELA) instead of the statistically more common *yhdisteestä* ('of compound' SG ELA). However, as with English, the POS information does fail to rerank cases where two equally rare word-forms with the same POS occur at the same edit distance, which seems to be common with participles, such as correcting *varustunut* to *varautunut* in stead of *varastanut*.

Furthermore we note that the the discourse particles that were dropped from the POS tagger's analysis tag set in order to decrease the tag set size will cause certain word forms in the dictionary to be incorrectly reranked, such as when correcting the very common misspelling *muillekkin* into *muillekokin* ('for others as well?' PL ALL QST KIN) instead of the originally correct *muillekin* ('for others as well' PL ALL KIN), since the analyses QST (for question enclitic) and KIN (for additive enclitic) are both dropped from the POS analyses.

4.3 *Performance Evaluation*

We did not work on optimizing the N-gram analysis and selection, but we found that the speed of the system is reasonable—even in its current form, considering that the algorithm is applied only to incorrect words on the user's request. Table 5 summarizes the average speed of performing the experiments in Table 4.

Table 5. The speed of ranking the errors.

Material Algorithm	English	Finnish
Unigram (baseline)	10.0 s	51.8 s
	399.1 wps	6.2 wps
POS N-grams	377.4 s	1616.2 s
	10.6 wps	0.14 wps

The performance penalty that is incurred on Finnish spell-checking but not so much on English comes from the method of determining readings for words unknown to the language model, i.e. from the guessing algorithm. The amount of words unknown to the language model in Finnish was greater than for English due to the training data sparseness and the morphological complexity of the language.

5 Future Work and Discussion

In this work we recreated the results of basic and context-based spelling correction for English and implemented same system for Finnish. We have shown that the POS based N-gram models are suitable for improving the spelling corrections for both morphologically more complex languages such as Finnish and for further improving languages with simpler morphologies like English. To further verify the claim, the method may still need to be tested on a typologically wider spectrum of languages.

In this article, we used readily available and hand-made error corpora to test our error correction method. A similar method as the one we use for error correction should be possible in error detection as well, especially when detecting real-word errors [7]. In future research, an obvious development is to integrate the N-gram system as a part of a real spell-checker system for both detection and correction of spelling errors, as is already done for the unigram based spell checker demonstrated in [2].

The article discussed only the reranking over basic edit distance error models, further research should include more careful statistical training for the error model as well, such as one demonstrated in [15].

6 Conclusion

In this paper we have demonstrated the use of finite-state methods for trigram based generation of spelling suggestions. We have shown that the basic word-form trigram methods suggested for languages like English do not seem to be as useful without modification for morphologically more complex languages like Finnish. Instead a more elaborate N-gram scheme using POS n-grams is successful for Finnish as well as English.

ACKNOWLEDGEMENTS

We are grateful to Sam Hardwick for making the spell checking software available and the HFST research group for fruitful discussions. We also thank the anonymous reviewers for useful suggestions and pointers.

REFERENCES

1. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. ACM (7) (1964)
2. Pirinen, T.A., Lindén, K.: Finite-state spell-checking with weighted language and error models. In: Proceedings of the Seventh SaLTMiL workshop on creation and use of basic lexical resources for less-resourced languagages, Valletta, Malta (2010) 13–18
3. Mitton, R.: Ordering the suggestions of a spellchecker without using context*. Nat. Lang. Eng. **15**(2) (2009) 173–192
4. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics—Doklady 10, 707–710. Translated from Doklady Akademii Nauk SSSR (1966) 845–848
5. Kukich, K.: Techniques for automatically correcting words in text. ACM Comput. Surv. **24**(4) (1992) 377–439
6. Church, K., Gale, W.: Probability scoring for spelling correction. Statistics and Computing (1991) 93–103
7. Mays, E., Damerau, F.J., Mercer, R.L.: Context based spelling correction. Inf. Process. Manage. **27**(5) (1991) 517–522
8. Silfverberg, M., Lindén, K.: Combining statistical models for pos tagging using finite-state calculus. In: Proceedings of the 18th Conference on Computational Linguistics, NODALIDA 2011. (2011) 183–190
9. Otero, J., Graña, J., Vilares, M.: Contextual spelling correction. In Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A., eds.: EUROCAST. Volume 4739 of Lecture Notes in Computer Science., Springer (2007) 290–296
10. Savary, A.: Typographical nearest-neighbor search in a finite-state lexicon and its application to spelling correction. In: CIAA '01: Revised Papers from the 6th International Conference on Implementation and Application of Automata, London, UK, Springer-Verlag (2002) 251–260
11. Mohri, M., Riley, M.: An efficient algorithm for the n-best-strings problem (2002)
12. Norvig, P.: How to write a spelling corrector. Web Page, Visited February 28th 2010, Available `http://norvig.com/spell-correct.html` (2010)
13. Pirinen, T.A.: Modularisation of finnish finite-state language description—towards wide collaboration in open source development of a morphological analyser. In Pedersen, B.S., Nešpore, G., Inguna Skadi n., eds.: Nodalida 2011. Volume 11 of NEALT Proceedings., NEALT (2011) 299—302

14. Hakulinen, A., Vilkuna, M., Korhonen, R., Koivisto, V., Heinonen, T.R., Alho, I.: Iso suomen kielioppi (2008) referred on 31.12.2008, available from `http://kaino.kotus.fi/visk`.
15. Brill, E., Moore, R.C.: An improved error model for noisy channel spelling correction. In: ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, Morristown, NJ, USA, Association for Computational Linguistics (2000) 286–293

TOMMI A. PIRINEN
DEPARTMENT OF MODERN LANGUAGES,
UNIVERSITY OF HELSINKI,
00014, FINLAND
E-MAIL: <TOMMI.PIRINEN@HELSINKI.FI>

MIIKKA SILFVERBERG
DEPARTMENT OF MODERN LANGUAGES,
UNIVERSITY OF HELSINKI,
00014, FINLAND
E-MAIL: <MIIKKA.SILFVERBERG@HELSINKI.FI>

KRISTER LINDÉN
DEPARTMENT OF MODERN LANGUAGES,
UNIVERSITY OF HELSINKI,
00014, FINLAND
E-MAIL: <KRISTER.LINDEN@HELSINKI.FI>

Author Index

Yuji Matsumoto
Diana McCarthy
Helen Meng
Rada Mihalcea
Ruslan Mitkov
Dunja Mladenic
Marie-Francine Moens
Masaki Murata
Vivi Nastase
Roberto Navigli
Kjetil Nørvåg
Constantin Orăsan
Patrick Saint-Dizier
Maria Teresa Pazienza
Ted Pedersen
Viktor Pekar
Anselmo Peñas
Stelios Piperidis
Irina Prodanof
Aarne Ranta
Victor Raskin
Fuji Ren
German Rigau
Fabio Rinaldi
Horacio Rodriguez
Vasile Rus
Horacio Saggion
Kepa Sarasola
Serge Sharoff
Grigori Sidorov
Thamar Solorio
John Sowa
Ralf Steinberger
Vera Lúcia Strube De Lima
Tomek Strzalkowski
Jun Suzuki
Christoph Tillmann
George Tsatsaronis
Junichi Tsujii
Dan Tufiş
Hans Uszkoreit
Felisa Verdejo
Manuel Vilares Ferro
Haifeng Wang
Bonnie Webber

ADDITIONAL REFEREES FOR THIS VOLUME

Adam Kilgarriff
Adrián Blanco González
Ahmad Emami
Akinori Fujino
Alexandra Balahur
Alvaro Rodrigo
Amitava Das
Ana Garcia-Serrano
Ananthakrishnan Ramanathan
Andrej Gardon
Aniruddha Ghosh
Antoni Oliver
Anup Kumar Kolya
Arantza Casillas-Rubio
Arkaitz Zubiaga
Bing Xiang
Binod Gyawali
Blaz Novak
Charlie Greenbacker
Clarissa Xavier
Colette Joubarne
Csaba Bodor
Daisuke Bekki
Daniel Eisinger
Danilo Croce
David Vilar

Delia Rusu
Diana Trandabat
Diman Ghazi
Dipankar Das
Egoitz Laparra
Ekaterina Ovchinnikova
Enrique Amigó
Eugen Ignat
Fazel Keshtkar
Feiyu Xu
Francisco Jose Ribadas Pena
Frederik Vaassen
Gabriela Ferraro
Gabriela Ramirez De La Rosa
Gerold Schneider
Gorka Labaka
Guenter Neumann
Guillermo Garrido
H M Ishrar Hussain
Håkan Burden
Hendra Setiawan
Hiroya Takamura
Hiroyuki Shindo
Ingo Glöckner
Ionut Cristian Pistol
Irina Chugur
Irina Temnikova
Jae-Woong Choe
Janez Brank
Jirka Hana
Jirka Hana
Jordi Atserias
Julian Brooke
K. V. S. Prasad
Katsuhito Sudoh
Kishiko Ueno
Kostas Stefanidis
Kow Kuroda
Krasimir Angelov
Laritza Hernández
Le An Ha
Liliana Barrio-Alvers
Lorand Dali
Luis Otávio De Colla Furquim
Luz Rello
Maite Oronoz Anchordoqui
Maria Kissa
Mario Karlovcec
Martin Scaiano
Masaaki Nagata
Matthias Reimann
Maud Ehrmann
Maya Carrillo
Michael Piotrowski
Miguel Angel Rios Gaona
Miguel Ballesteros
Mihai Alex Moruz
Milagros Fernández Gavilanes
Milos Jakubicek
Miranda Chong
Mitja Trampus
Monica Macoveiciuc
Najeh Hajlaoui
Natalia Konstantinova
Nathan Michalov
Nattiya Kanhabua
Nenad Tomasev
Niyu Ge
Noushin Rezapour Asheghi
Oana Frunza
Oier Lopez De Lacalle
Olga Kolesnikova
Omar Alonso
Paolo Annesi
Peter Ljunglöf
Pinaki Bhaskar
Prokopis Prokopidis
Rainer Winnenburg
Ramona Enache
Raquel Martínez

Richard Forsyth
Robin Cooper
Rodrigo Agerri
Roser Morante
Ryo Otoguro
Samira Shaikh
Santanu Pal
Shamima Mithun
Sharon Small
Simon Mille
Simone Paolo Ponzetto
Siva Reddy
Somnath Banerjee
Tadej Štajner
Thierry Declerck
Ting Liu
Tom De Smedt
Tommaso Caselli
Tong Wang
Toshiyuki Kanamaru
Tsutomu Hirao
Ulf Hermjakob
Upendra Sapkota
Vanessa Murdock
Victor Darriba
Víctor Peinado
Vít Baisa
Vojtech Kovar
Wilker Aziz
Yulia Ledeneva
Yvonne Skalban
Zuzana Neverilova