Volume 2 ▪ Issue 1 ▪ August 2011

INTERNATIONAL JOURNAL OF

COMPUTATIONAL LINGUISTICS (IJCL)

Publication Frequency: 6 Issues / Year

ISSN : 2180-1266

CSC PUBLISHERS

http://www.cscjournals.org

INTERNATIONAL JOURNAL OF COMPUTATIONAL LINGUISTICS (IJCL)

VOLUME 2, ISSUE 1, 2011

EDITED BY
DR. NABEEL TAHIR

ISSN (Online): 2180 - 1266

International Journal of Computational Linguistics (IJCL) is published both in traditional paper form and in Internet. This journal is published at the website http://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJCL Journal is a part of CSC Publishers
Computer Science Journals
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF COMPUTATIONAL LINGUISTICS (IJCL)

Book: Volume 2, Issue 1, August 2011
Publishing Date: 31-08-2011
ISSN (Online): 2180-1266

IJCL Journal is a part of CSC Publishers
http://www.cscjournals.org

© IJCL Journal
Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers, 2011

EDITORIAL PREFACE

The International Journal of Computational Linguistics (IJCL) is an effective medium for interchange of high quality theoretical and applied research in Computational Linguistics from theoretical research to application development. This is the third issue of second volume of IJCL. The Journal is published bi-monthly, with papers being peer reviewed to high international standards. International Journal of Computational Linguistics (IJCL) publish papers that describe state of the art techniques, scientific research studies and results in computational linguistics in general but on theoretical linguistics, psycholinguistics, natural language processing, grammatical inference, machine learning and cognitive science computational models of linguistic theorizing: standard and enriched context free models, principles and parameters models, optimality theory and researchers working within the minimalist program, and other approaches.

IJCL give an opportunity to scientists, researchers, and vendors from different disciplines of Artificial Intelligence to share the ideas, identify problems, investigate relevant issues, share common interests, explore new approaches, and initiate possible collaborative research and system development. This journal is helpful for the researchers and R&D engineers, scientists all those persons who are involve in Computational Linguistics.

Highly professional scholars give their efforts, valuable time, expertise and motivation to IJCL as Editorial board members. All submissions are evaluated by the International Editorial Board. The International Editorial Board ensures that significant developments in image processing from around the world are reflected in the IJCL publications.

IJCL editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the Journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Scribd, CiteSeerX Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJCL. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJCL provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts.

Editorial Board Members
International Journal of Computational Linguistics (IJCL)

EDITORIAL BOARD

TABLE OF CONTENTS

Volume 2, Issue 1, August 2011

Implementation of Enhanced Parts-of-Speech Based Rules for English to Telugu Machine Translation

A. P. Siva Kumar *sivakumar.ap@gmail.com*
Assistant Professor,
Department of Computer Science and Engineering
JNTUA College of Engineering, Anantapur-516390, India.

Dr. P. Premchand *p.premchand@uceou.edu*
Professor,Department of Computer Science and Engineering
Osmania University,Hyderabad, India.

Dr. A. Govardhan *govardhan_cse@yahoo.co.in*
Principal & Professor,
Department of Computer Science and Engineering
JNTUH College of Engineering,Nachupalli, India.

Abstract

Words of a sentence will not follow same ordering in different languages. This paper proposes certain Parts-of-Speech (POS) based rules for reordering the given English sentence to get translation in Telugu. The added rules for adverbs, exceptional conjunctions in addition to improved handling of inflections enable the system to achieve more accurate translation. The proposed rules along with existing system gave a score of 0.6190 with BLEU evaluation metric while translating sentences from English to Telugu. This paper deals with simple form of sentences in a better way.

Keywords: POS-based Reordering, English to Telugu CLIR, BLEU

1. INTRODUCTION

Information Retrieval (IR) refers to the extraction of required information with a user query (formal statement of information need) written in one language (source language), from a large repository of documents that may be written in the same or some other language (target language). Getting only relevant data from the existing literature is made easy and faster by IR systems. The ever increasing requirement for multi-lingual information access along with the lack of technical support for multi-lingual processing bring about a new branch in research of Information Retrieval named Cross Language Information Retrieval (CLIR). It makes use of user queries written in one language to retrieve the relevant documents written in some other language. For example, a user may pose their query in English but retrieve relevant documents written in French.

English (source language) is a Subject-Verb-Object patterned language whereas Telugu (target language) is a Subject-Object-Verb patterned language that is the order of words with different parts-of-speech (POS) is not same in source and target languages. So, when a sentence is translated from source language to target language using word to word translation, the meaning of the sentence might be lost. This problem can be solved by reordering the words in the sentence based on some POS based rules.
POS tagger tool is used to identify the parts-of-speech of each word in the sentence. Then certain rules proposed in this paper, can be applied on the source sentence followed by word to word dictionary based translation. Gender based inflections are also handled. The added features enhance the quality of translated sentence by giving more accurate meaning.

The paper is organized as follows. Section 2 outlines the previous work on the translation by various organizations. Section 3 explains about the proposed system in detail. Section 4 contains the experimental results obtained by using this system and Section 5 concludes the paper.

2. PREVIOUS WORK

CLIR for Indian languages is undergoing considerable amount of research in various universities herein like Indian Institute of India (IIT), Bombay; National Centre or Software Technology (NCST) Mumbai (now, Centre for Development of Advanced Computing (CDAC), Bombay; International Institute of Information Technology (IIIT), Hyderabad. There are many machine translator systems still under production in India such as Anusaaraka project being done by IIIT, Hyderabad; Mantra (MAchine assisted TRAnslation tool) that converts English text into Hindi in a precise domain of personal administration, office orders, etc.; AnglaBharti project that is based on Pseudo Lingua for Indian Languages (PLIL). Reference [2] proposes several linguistic rules that could be incorporated in Generalized Example Based Machine Translation (G-EBMT) system for translation of English to any of the Indian languages like Telugu, Kannada, Malayalam and Tamil. The concept of word reordering of the source language sentence based on parts-of-speech tags is used also in Reference [4] for the languages Spanish, German and English.

The existing system uses generalized example based machine translation along with some linguistic rules that guide reordering of words present in a source language sentence. The dictionary based word to word translation will be the next step after reordering to achieve desired target language sentence.

3. PROPOSED SYSTEM

The design of the proposed system is an extension to the existing systems for reordering. Various stages are followed while translating a sentence from source language to target language. In each stage various reordering rules are applied to get a target sentence with correct meaning.

This system reorders the given sentence by first dividing it into words and attaching tags by using the POS tagger mentioned in [11]. Then the rules mentioned below will be applied to reorder the sentence.

3.1 Existing Rules

3.1.1 Verb Rule

This rule deals with the sentences consisting of a verb. If verb is present in the sentence, it should be moved to the end.

Consider “I eat mango” (English). This will be reordered as “I mango eat” as “eat” is a verb. Its translation will be “nenu maamidipandu tintaanu” (Telugu).

3.1.2 Conjunction Rule

It can handle sentences with one conjunction which may be present at the beginning or in the middle of the sentence. The parts of the sentence before and after the conjunction are treated as separate phrases which are translated separately and joined at the end in the same order. Consider “I studied well but the results are poor” (English). Here “I studied well” and “the results are poor” are considered as two phrases separated by the conjunction “but”. So, the two phrases are translated separately and joined at the end as “Nenu baaga chadivaanu kani manchi phalitalu raledu” (Telugu).

3.2 Proposed Rules

3.2.1 Proper Noun Rule

This rule deals with proper noun that refers to name of a company, organization, institute, person etc. which cannot be translated. In such case we use transliteration directly.

Consider ramu, john, jntu, IBM etc. Here, these words will be transliterated as they cannot be translated using dictionary. Other types of nouns (e.g. cow, chair, banana) can be translated directly using dictionary.

3.2.2 Adverb Rule
Sentences having verb and adverb should be reordered in such a way that verb is placed at the end of phrase immediately preceded by adverb.

Consider “He walks faster than Rajesh” (English). Here “walks” is verb and “faster” is adverb. Its translation will be “Atadu Rajesh kanna tvaraga nadustadu” (Telugu). Here “nadustadu” (verb) is placed at the end immediately preceded by “tvaraga” (adverb).

3.2.3 Dative Rule
This rule deals with a noun or pronoun when it is the indirect object (refers to the person or thing that an action is done to or for) of a verb. Indirect object is appended with either “ku” or “kosam” accordingly while translation.

Consider “He gave her a gift” (in English). This should be translated as “Ameku athadu oka bahumanam ichadu”. Here “her” is an indirect object. When word to word translation is performed, “her” is translated to “ame”. But, it does not give correct meaning. So, by applying this rule, we get translation as “ameku”.

3.2.4 Conjunction Exception Rule
This handles exceptional cases of conjunction rule. It says that the phrases of a sentence having conjunctions like “if”, “though” and “although” should be swapped as they will take different ordering in English and Telugu.

Consider “You will pass the exam if you study well” (English). Here the phrases are “you will pass the exam” and “you study well” should be swapped and translated as “nuvvu baaga chadivithe nuvvu pareekshalu paasavuthaavu” (Telugu).

3.3 Stages of Translation
The above mentioned rules for translation can be performed by applying them in a specific order as explained below (as shown in Figure1)

3.3.1 Stage 1
Initially, a POS tagger tool is used to associate each word in the sentence with the corresponding parts-of-speech tags. Based on the tag linked with each word the reordering is performed. For much better translation a better tagger can be used.

For example, “Rajesh walks fast but he failed in the competition.” is tagged by the POS tagger as:
Rajesh_NNP walks_VBZ fast_RB but_CC he_PRP failed_VBD in_IN the_DT competition_NN.
Here, NNP-Singular or mass noun,
VBZ - verb, 3rd. singular present,
RB-Adverb,
CC- Coordinating Conjunction,
PRP- singular nominative pronoun,
VBD-past tense verb,
IN - Preposition or subordinating conjunction,
DT- singular determiner/quantifier and
NN - Noun, singular or mass

3.3.2 Stage 2
In this stage, the presence of conjunction is checked. If it is not present then the flow is directly transferred to stage3. Else, the conjunction rule is applied. The exception with the conjunctions is

also handled in this stage. If the exception case occurs with the conjunction, the sentence is reordered accordingly by applying conjunction exception rule.
For example, “Rajesh walks fast but he failed in the competition”. Here firstly the presence of conjunction is checked. The conjunction “but” is present, so the sentence is divided into three phrases

p1: Rajesh walks fast
p2: but
p3: he failed in the competition

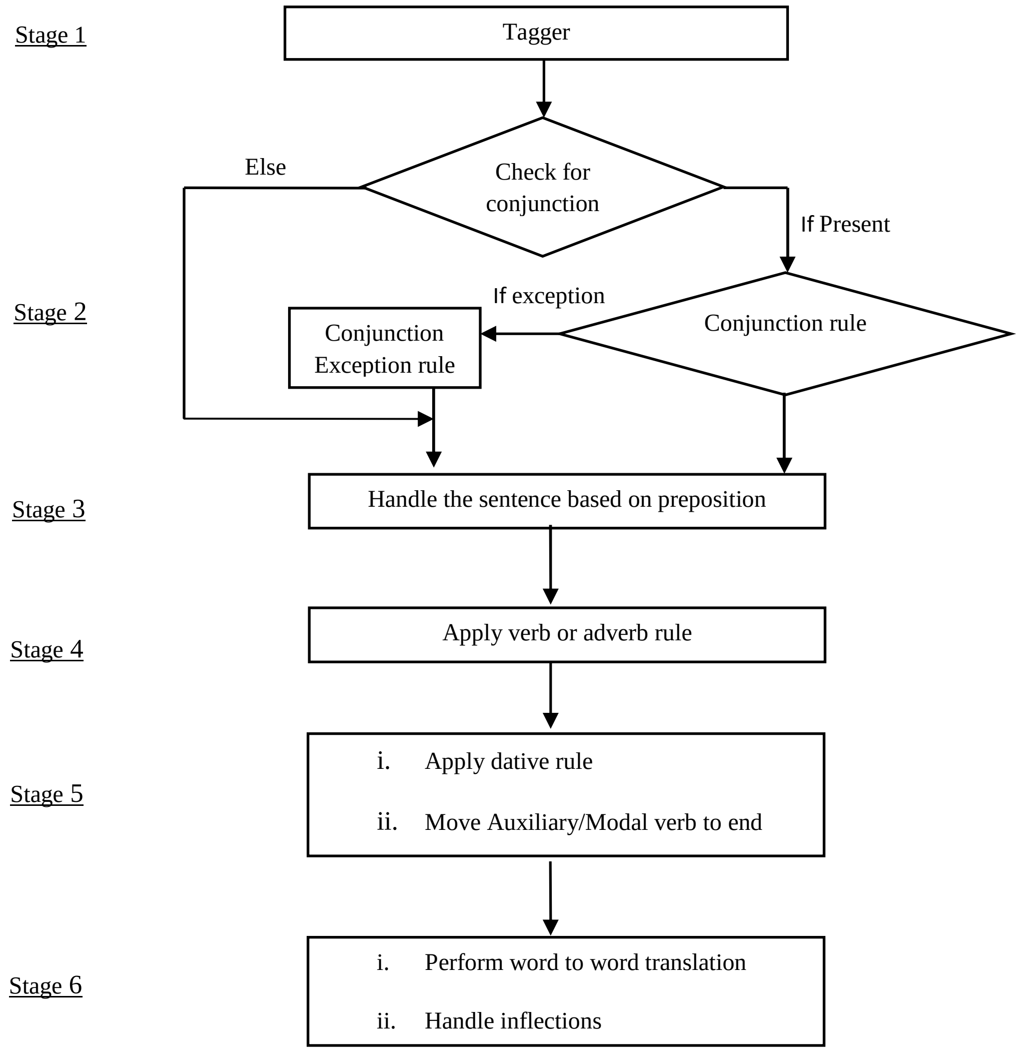

Stage 1
Tagger
Else
Check for conjunction
If Present
If exception
Stage 2
Conjunction Exception rule
Conjunction rule
Stage 3
Handle the sentence based on preposition
Stage 4
Apply verb or adverb rule
Stage 5
i. Apply dative rule
ii. Move Auxiliary/Modal verb to end
Stage 6
i. Perform word to word translation
ii. Handle inflections

FIGURE 1: Flow of operations in translation from source language to target language.

All the reordering rules are applied separately for the two phrases (p1 and p3). While checking the presence of conjunction, it is also verified that whether it is an exceptional conjunction or not. If so, it is handled separately by swapping the phrases before and after the conjunction. Consider ‘You will pass the exam if you study well’. The sentence contains the ‘if’ exceptional conjunction. So the phrases should be reordered as shown in Table 1.

Sentence	Phrases
You will pass the exam <u>if</u> you study well (Original sentence)	p1: You will pass the exam p2: if p3: you study well
You study well <u>if</u> you will pass the exam (After Reordering)	p1: you study well p2: if p3: you will pass the exam

TABLE 1: Sentence with and without conjunction rule

3.3.3 Stage 3

Here, the sentence is split based on the preposition present in it. Then the phrases before and after the preposition are swapped.
p1: Rajesh walks fast
p2: but
p3: he failed in the competition

For the above example, the preposition is present only in the p3 phrase. So p3 should be split as p3 and p4. After reordering the phrases are as follows:
p1: Rajesh walks fast
p2: but
p3: the competition in
p4: he failed

3.3.4 Stage 4

In this stage, the presence of verb or the combination of adverb and verb is checked and verb rule or adverb rule are applied accordingly. For the above example, p1 has the combination of verb and adverb and hence they are reordered as
p1: Rajesh fast walks
p2: but
p3: the competition in
p4: he failed

3.3.5 Stage 5

Here, the dative cases are checked and if present, dative rule is applied. And also in this stage, the auxiliary/modal verbs are identified. If an auxiliary/modal verb is present in any of the parts, it will be placed at the end of that phrase.

Consider an example “he is playing games”. After crossing the above stages the sentence will be “he is games playing”. Here “is” is an auxiliary verb, thus it should be moved to the end of the sentence as “he games playing is”.

3.3.6 Stage 6

After crossing all the above 5 stages the word to word translation is performed by using bilingual English to Telugu dictionary. Then Proper noun rule is applied for the words not found in dictionary. This stage also handles the inflections that are different forms of a verb based on the gender after translation into target language.

For the above example “Ramesh” is not found in the dictionary so the proper noun rule is applied and the translated phrases will be
p1: Ramesh veganga nadu
p2: kani
p3: poti lo
p4: athadu viphalam ayyenu

Also the inflections present in the sentence will be handled as given in [2]. Thus at the end, combining all the phrases with the inflection rule we get the translated sentence as:
'Ramesh veganga nadustadu kani poti lo athadu viphalam ayyenu'
In this way by following all the six stages an English sentence can be translated to Telugu appropriately giving a better quality translation.

4. EXPERIMENT

For the evaluation of the proposed system we have selected 100 simple English sentences from the daily newspaper in which the count of words varies from 3 to 12. For translation purpose, we have used a bilingual dictionary containing all the words used in testing corpus. To perform evaluation technique the sentences are translated by the proposed system and also by a human.

Quality can be treated as the agreement between the machine translation and the human translation. The system is said to be good if its translation is very close to that of the human translation. To determine this quality of the proposed system we used BLEU (Bilingual Evaluation Understudy) score evaluation technique referred in [3]. The BLEU score is given by,

$$BLEU = BP.\exp\left(\textstyle\sum_n w_n \log p_n\right) \qquad \text{(i)}$$

Here,

$$p_n = \frac{count\ of\ correct\ n\text{-}gram\ match}{count\ of\ total\ n\text{-}gram}$$

where BP is the brevity penalty factor, given by,

$$BP = \begin{cases} 1 & if\ c > r \\ e^{(1-r/c)} & if\ c \le r \end{cases}$$

w_n = positive weights = 1/N,
p_n = modified n gram precisions,
c = length of the translation obtained from the system,
r = length of the correct translation translated by a human.

Applying log to (i),

$$\log_e BLEU = \min(1 - r/c, 0) + \sum_n w_n \log p_n$$

In the proposed system the length of the sentence starts from 3. Hence we use N=3 (that is trigram model) in the system. The trigram model consists of subsequence of 3 words to form trigrams. By examining how many standard deviations each 3-gram differs from its mean occurrence, the p_n value is determined. The evaluation technique when performed on proposed system with a set of 100 sentences gave a score of 0.6190.

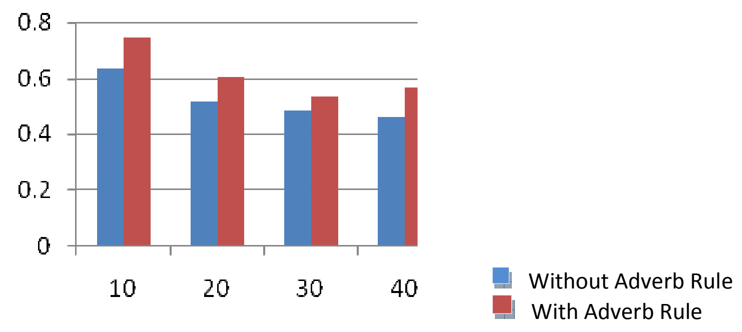


FIGURE 2: BLEU score without and with adverb rule

The values in the Table 2 can be represented as Figure 2, which shows the BLEU scores of sentences without adverb rule against with adverb rule for English to Telugu translation. The x-axis represents the number of sentences and the y-axis represents the BLEU score.

No. of sentences	Without adverb rule	With adverb rule
10	0.6364	0.7444
20	0.5161	0.6040
30	0.4830	0.5357
40	0.4637	0.5687

TABLE 2: BLEU score for without and with adverb rule

In the similar way, Figure 3 shows the BLEU scores of sentences without conjunction exception rule against with conjunction exception rule for English to Telugu translation, which are tabulated in Table 3. In this figure also the x-axis represents the number of sentences and the y-axis represents the BLEU score.

No. of sentences	Without conjunction exception rule	With conjunction exception rule
10	0.2024	0.2393
20	0.2650	0.2864
30	0.2522	0.2666
40	0.2251	0.2619

TABLE 3: BLEU score for without and with conjunction exception rule

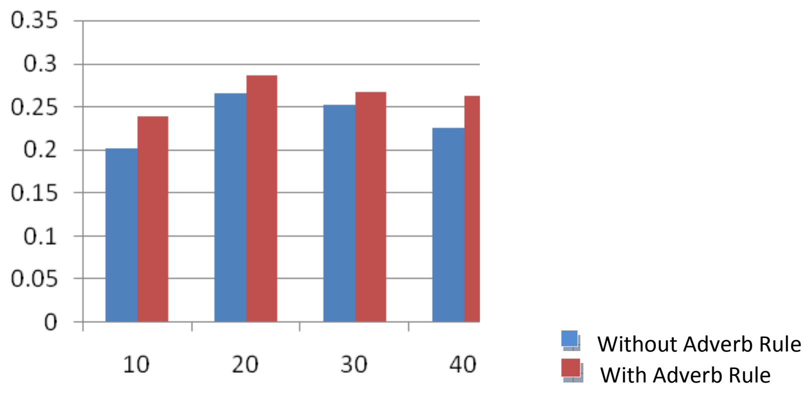


FIGURE 3: BLEU score without and with conjunction exception rule

5. CONCLUSION AND FUTURE WORK

This paper enhances POS based reordering rules that preprocess the user query for better translation in order to use it in searching relevant documents written in Telugu. The added rules enable the system to deal with adverbs and conjunctions in a better way. The proposed system gives a BLEU score of 0.6190 (on an average). The performance of the system highly depends on the POS tags attached to the given source sentence. Better the tagger, the more efficient the translation will be.

There is no perfect machine translator for Indian languages which stem from Sanskrit and Dravidian family, mainly because of the reason that they are rich in sandhis. More concentration should be given to handle this. We also would like to handle other type of sentences like interrogations and exclamations in future work.

6. REFERENCES

[1] R.Gangadharaiah & N. Balakrishnan, "*Application of Linguistic Rules to Generalized Example Based Machine Translation for Indian Languages*", Proceedings of the First National Symposium on Modeling and Shallow Parsing of Indian Languages, India, 2006

[2] Mustafa Abusalah, John Tait & Michael Oakes, "*Literature Review of Cross Language Information Retrieval*", World Academy of Science, Engineering and Technology, 2005.

[3] P.Kishore, Salim Roukas, Todd ward & Wei-Jing Zhu, "*BLEU: a Method for Automatic Evaluation of Machine Translation*", Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, pp. 311-318, 2002.

[4] Maja Popovic & Hermann Ney, "*POS-based Word Reorderings for Statistical Machine Translation*", in Proceedings of the Fifth International conference on Language Resources and Evaluation, 2006.

[5] Anne R. Diekema, "*Translation Events in Cross-Language Information Retrieval: Lexical Ambiguity, Lexical Holes, Vocabulary Mismatch, and Correct Translation*", Dissertation at School of Information Studies, Syracuse University, 2003.

[6] Sethuramalingam S, "*Effective Query Translation Techniques for Cross-Language Information Retrieval*", MS Thesis submitted at IIIT Hyderabad, India, 2009.

[7] Sudip Naskar & Sivaji Bandyopadhyay, "*Use of Machine Translation in India: Current Status*", AAMT J., 36:25-31, 2004.

[8] Sanjay Kumar Dwivedi and Pramod Premdas Sukhdeve, "*Machine Translation System in Indian Perspectives*", Journal of Computer Science 6 (10): 1082-1087, 2010.

[9] Shu Cai, Yajuan L & Qun Liu, "*Improved Reordering Rules for Hierarchical Phrase-based Translation*", International Conference on Asian Language Processing, 2009.

[10] ZHANG Xiao-fei, HUANG He-yan & ZHANG Ke-liang, "*Cross-Language Information Retrieval Based on Weight Computation of Query Keywords Translation*", Intelligent Computing and Intelligent Systems, 2009 IEEE International Conference, 2009.

[11] Parts-Of-Speech tagger tool – http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/postagger.

Named Entity Recognition System for Hindi Language: A Hybrid Approach

Shilpi Srivastava *shilpii26@gmail.com*
Department of Computer Science
University of Mumbai, Vidyanagri, Santacruz (E)
Mumbai-400098, India

Mukund Sanglikar *masanglikar@rediffmail.com*
Professor, Department of Mathematics,
Mithibai college, Vile Parle (W), University of Mumbai
Mumbai-400056, India

D.C Kothari *kothari@mu.ac.in*
Professor, Department of Physics,
University of Mumbai, Vidyanagri, Santacruz(E)
Mumbai-400098, India

Abstract

Named Entity Recognition (NER) is a major early step in Natural Language Processing (NLP) tasks like machine translation, text to speech synthesis, natural language understanding etc. It seeks to classify words which represent names in text into predefined categories like location, person-name, organization, date, time etc. In this paper we have used a combination of machine learning and Rule based approaches to classify named entities. The paper introduces a hybrid approach for NER. We have experimented with Statistical approaches like Conditional Random Fields (CRF) & Maximum Entropy (MaxEnt) and Rule based approach based on the set of linguistic rules. Linguistic approach plays a vital role in overcoming the limitations of statistical models for morphologically rich language like Hindi. Also the system uses voting method to improve the performance of the NER system.

Keywords: NER, MaxEnt, CRF, Rule base, Voting, Hybrid Approach

1. INTRODUCTION

Named Entity Recognition is a subtask of Information extraction where we locate and classify proper names in text into predefined categories. NER is a precursor for many natural languages processing tasks. An accurate NER system is needed for machine translation, more accurate internet search engines, automatic indexing of documents, automatic question-answering, information retrieval etc

Most NER systems use a rule based approach or statistical machine learning approach or a combination of these. A Rule-based NER system uses hand-written rules to tag a corpus with named entity (NE) tags. Machine-learning (ML) approaches are popularly used in NER because these are easily trainable, adaptable to different domains and languages and their maintenance is less expensive. A hybrid NER system is a combination of both rule-based and statistical approaches.

Not much work has been done on NER for Indian languages like Hindi. Hindi is the third most spoken language of the world and still no accurate Hindi NER system exists. As some features like capitalization are not available in Hindi and due to lack of a large labeled dataset and of standardization and spelling variations, an English NER system cannot be used directly for Hindi. There is a need to develop an accurate Hindi NER system for better presence of Hindi on the internet. It is necessary to understand Hindi language structure and learn new features for building better Hindi NER systems.

In this paper, we have reported a NER system for Hindi by using the classifiers, namely MaxEnt, CRF and Rulebase model. We have demonstrated a comparative study of performance of the two statistical classifiers (MaxEnt & CRF) widely used in NLP tasks, and use a novel voting mechanism based on classification confidence (that has a statistical validity) to combine the two classifiers among with preliminary handcrafted rules.

Our proposed system is an attempt to illustrate the hybrid approach for Hindi Named Entity Recognition. The system makes use of some POS information of the words along with the variety of orthographic word level features that are helpful in predicting the various NE classes. Theoretically it is known that CRF is better than MaxEnt due to the label bias problem of MaxEnt. The main contribution of this work is to make a comparative study between the two classifiers MaxEnt and CRF and Results show that CRF always gave better results in comparison to MaxEnt.

In the following sections, we will discuss about previous works, the issues in Hindi language & various approaches for NER task and examine our approach, design and implementation details, results and concluding discussion.

2. RELATED WORKS

NER has drawn more and more attention from NLP researchers since the last decade (Chinchor 1995, Chinchor 1998) [5] [18]. Two generally classified approaches to NER are Linguistic approach and Machine learning (ML) based approach. The Linguistics approach uses rule-based models manually written by linguists. ML based techniques make use of a large amount of annotated training data to acquire high-level language knowledge. Various ML techniques which are used for the NER task are Hidden Markov Model (HMM) [7], Maximum Entropy Model (MaxEnt) [6], Decision Tree [3], Support Vector Machines [4] and Conditional Random Fields (CRFs) [10]. Both the approaches may make use of gazetteer information to build system because it improves the accuracy.

Ralph Grishman in 1995 developed a rule-based NER system which uses some specialized name dictionaries including names of all countries, names of major cities, names of companies, common first names etc [15]. Another rule-based NER system is developed in 1996 which make use of several gazetteers like organization names, location names, person names, human titles etc [16]. But the main disadvantages of these rule based techniques are that these require huge experience and grammatical knowledge of particular languages or domains and these systems are not transferable to other languages.

Here we mention a few NER systems that have used ML techniques. 'Identifinder' is one of the first generation ML based NER systems which used Hidden Markov Model (HMM) [7]. By using mainly capital letter and digit information, this system achieved F-value of 87.6 on English. Borthwick used MaxEnt in his NER system with lexical information, section information and dictionary features [6]. He had also shown that ML approaches can be combined with hand-coded systems to achieve better performance. He was able to develop a 92% accurate English NER system. Mikheev et al. has also developed a hybrid system containing statistical and hand coded system that achieved F-value of 93.39 [17].

Other ML approaches like Support Vector Machine (SVM), Conditional Random Field (CRF), and Maximum Entropy Markov Model (MEMM) are also used in developing NER systems. Combinations of different ML approaches are also used. For example, we can mention a system developed by Srihari et al., which combined several modules, built by using MaxEnt, HMM and handcrafted rules, that achieved F-value of 93.5 [19].

The NER task for Hindi has been explored by Cucerzan and Yarowsky in their language independent NER which used morphological and contextual evidences [20]. They ran their experiments with 5 languages: Romanian, English, Greek, Turkish and Hindi. Among these, the accuracy for Hindi was the worst. A Recent Hindi NER system is developed by Li and McCallum using CRF with feature induction [21]. They automatically discovered relevant features by providing a large array of lexical tests and using feature induction to automatically construct the features that mostly increase conditional likelihood. However the performance of these systems is significantly hampered when the test corpus is not similar to the training corpus. Few studies (Guo et al., 2009), (Poibeau and Kosseim, 2001) have been performed towards genre/domain adaptation. But this still remains an open area. In IJCNLP-08 workshop on NER for South and South East Asian languages, held in 2008 at IIIT Hyderabad, was a major attempt in introducing NER for Indian languages that concentrated on five Indian languages- Hindi, Bengali, Oriya, Telugu and Urdu. As part of this shared task, [22] reported a CRF-based system followed by post-processing which involves using some heuristics or rules. Some efforts for Indian Language have also been made [23 [24]. A CRF-based system has been reported in [25], where it has been shown that the hybrid CRF based model can perform better than CRF. [26] presents a hybrid approach for identifying Hindi names, using knowledge infusion from multiple sources of evidence.

The authors, to the best of their knowledge and efforts have not encountered a work which demonstrates a comparative study between the two classifiers MaxEnt and CRF and uses a hybrid model based on MaxEnt, CRF and Rulebase for Hindi Named Entity Recognition.

3. ISSUES WITH HINDI LANGUAGE

The task of building a named entity recognizer for Hindi language presents several issues related to their linguistic characteristics. There are some issues faced by Hindi and other Indian languages:

- No capitalization: Unlike English and most of the European languages, Indian languages lack the capitalization information that plays a very important role to identify NEs in those languages. Hence English NER systems can exploit the feature of capitalization to its advantage because all English names always start with capital letters while Hindi names don't have scripts with graphical cues like capitalization, which could act as an important indicator for NER.

- Ambiguous names: Hindi names are ambiguous and this issue makes the recognition a very difficult task. One of the features of the named entities in Hindi language is the high overlap between common nouns and proper nouns. Indian person names are more diverse compared to those of most other languages and a lot of them can be found in the dictionary as common nouns.

- Scarcity of resources and tools: Hindi, like other Indian languages, is also a resource poor language. Annotated corpora, name dictionaries, good morphological analyzers, POS taggers etc. are not yet available in the required quantity and quality.

- Lack of standardization and spelling: Another important language related issue is the variation in the spellings of proper names. This increases the number of tokens to be learnt by the machine and would perhaps also require a higher level task like co-occurrence resolution.

- Free word order language: Indian languages have relatively free word order.
- Web sources for name lists are available in English, but such lists are not available in Indian languages.
- Although Indian languages have a very old and rich literary history still technology development are recent.
- Indian languages are highly inflected and provide rich and challenging sets of linguistic and statistical features resulting in long and complex word forms.
- Lack of labeled data.
- Non-availability of large gazetteer:

4. VARIOUS APPROACHES FOR NER

There are three basic approaches to NER [1]. They are rule based approach, statistical or machine learning approach and hybrid approach.

4.1 Rule Based Approach

It uses linguistic grammar-based techniques to find named entity (NE) tags. It needs rich and expressive rules and gives good results. It requires great knowledge of grammar and other language related rules. Good experience is needed to come up with good rules and heuristics. It is not easily portable and has high acquisition cost. It is very specific to the target data.

4.2 Statistical Methods or Machine Learning Methods

The common machine learning models used for NER are:

- **HMM [14]:** HMM stands for Hidden Markov Model. HMM is a generative model. The model assigns the joint probability to paired observation and label sequence. Then the parameters are trained to maximize the joint likelihood of training sets.

  It is advantageous as its basic theory is elegant and easy to understand. Hence it is easier to implement and analyze. It uses only positive data, so they can be easily scaled.

  It has few disadvantages. In order to define joint probability over observation and label sequence HMM needs to enumerate all possible observation sequence. Hence it makes various assumptions about data like Markovian assumption i.e. current label depends only on the previous label. Also it is not practical to represent multiple overlapping features and long term dependencies. Number of parameter to be evaluated is huge. So it needs a large data set for training.

- **MaxEnt [6]:** MaxEnt stands for Maximum Entropy Markov Model (MEMM). It is a conditional probabilistic sequence model. It can represent multiple features of a word and can also handle long term dependency. It is based on the principle of maximum entropy which states that the least biased model which considers all know facts is the one which maximizes entropy. Each source state has a exponential model that takes the observation feature as input and output a distribution over possible next state. Output labels are associated with states.

  It solves the problem of multiple feature representation and long term dependency issue faced by HMM. It has generally increased recall and greater precision than HMM.

  It also has some disadvantages. It has Label Bias Problem. The probability transition leaving any given state must sum to one. So it is biased towards states with lower outgoing transitions. The state with single outgoing state transition will ignore all observations. To handle Label Bias Problem we can change the state-transition.

- **CRF [10]:** CRF stands for Conditional Random Field. It is a type of discriminative probabilistic model. It has all the advantages of MEMMs without the label bias problem. CRFs are undirected graphical models (also know as random field) which is used to calculate the conditional probability of values on assigned output nodes given the values assigned to other assigned input nodes..

4.3 Hybrid Models

Hybrid models are basically combination of rules based and statistical models. In Hybrid NER system, approach uses the combination of both rule-based and ML technique and makes new methods using strongest points from each method. It is making use of essential feature from ML approaches and uses the rules to make it more efficient.

5. OUR APPROACH

5.1 CRF Based Machine Learning

The basis idea of CRF is to construct a conditional probability $P(Y \mid X)$ from the label sequence Y (e.g. NE tags) and observation sequence X (e.g. words) after model is constructed, then testing can be done by ending the label that maximizes $P(Y \mid X)$ for the observed features.

Definition [10]: " Let $G=(V,E)$ be a graph such that $Y=(Y_v)v \in V$, so that Y is indexed by the vertices of G. Then (X,Y) is a conditional random field in case, when conditioned on X , the random variables Y_v obey the Markov Property with respect to the graph: $P(Y_v \mid X, Y_w, w \neq v) = P(Y_v \mid X, Y_w, w \sim v)$; where w ~ v means that w and v are neighbors in G."

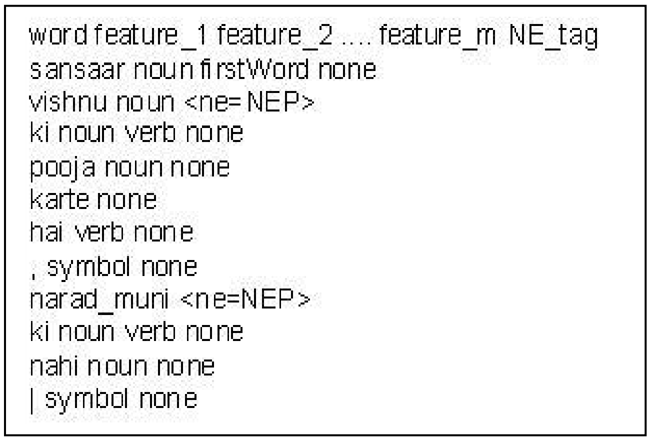
“Lafferty et. al [10] define the probability of a particular label sequence Y given the observation sequence X to be a normalized product of potential functions each of the form,

$$\exp\left(\sum_j \lambda_j t_j (y_{i-1}, y_i, x, i) + \sum_k \mu_k s_k (y_i, x, i) \right)$$

Where $t_j (y_{i-1}, y_i, x, i)$ is a transition feature function of the entire observation sequence and the labels at positions i and i-1 in the label sequence; $s_k (y_i, x, i)$ is a state feature function of the label at position i and the observation sequence; and λ_j and μ_k are parameters to be estimated from training data.

Final expression of probability of a label sequence Y given an observation sequence X is $p(y \mid x, \lambda) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{n} \sum_j \lambda_j f_i (y_{i-1}, y_i, x, i) \right)$ Where $f_i (y_{i-1}, y_i, x, i)$ is either a state function $s(y_{i-1}, y_i, x, i)$ or a transition function $t(y_{i-1}, y_i, x, i)$.” [13]

We are using mallet-0.4 [12] for training and testing. Mallet provides SimpleTagger program that takes input as a file in mallet format of Figure 1. After training the model is saved in a file. Then model file can be used for testing. When trained model is tested, it produces an output file that contains the predicted tags of the word. The predicted tags are present in the same line number as the text file.



```
word feature_1 feature_2 .... feature_m NE_tag
sansaar noun firstWord none
vishnu noun <ne=NEP>
ki noun verb none
pooja noun none
karte none
hai verb none
, symbol none
narad_muni <ne=NEP>
ki noun verb none
nahi noun none
| symbol none
```

FIGURE 1: Data in mallet format

5.2 MaxEnt Based Machine Learning

It is based on the principle of maximum entropy which states that the least biased model which considers all know facts is the one which maximizes entropy.

Let H be the set of histories and T be the set of allowable tags.
The maximum entropy model is defined over $H \times T$.
The model's probability is defined as probability of history h with tag.

$$p(h,t) = \pi\mu \prod_j \alpha_j^{f_i(h,t)}$$

Where,
π is normalization constant
μ, α_j are model parameters
$f_i(h,t)$ feature function
Let $L(p)$ = likelihood of training data using distribution,

$$L(p) = \prod_{i=1}^{n} p(h_i, t_i)$$

The method is to choose the model parameters correctly with respect to maximum likelihood principle.

We are using mallet-0.4 MaxEnt implementation. For the purpose of training and testing using MaxEnt, we created file MaxEntTagger which converts the input file in format specified in Figure 1 into their internal data structure. The file is similar to SimpleTagger. Then the training and testing is done similar to CRF.

5.3 Rule Based Model

Following rules were used to get NE tags from words

- <ne=NEN>: For numbers written in Hindi font like ek, paanch etc, word matching with dictionary is used. The file contain Hindi number words are provided by Hindi Wordnet [11]. If the number contains only digits then it is NEN.

- <ne=NEL>: Use dictionary matching for common locations like Bharat(India), Kanpur. Also used suffix matching like words ending with "pur" are generally cities like Kanpur, Nagpur, Jodhpur etc.
- <ne=NEB>: Used dictionary matching.
- <ne=NETI>: Used regular expression matching e.g. 12-3-2008 format is NETI
- <ne=NEP>: Suffix matching is used with common surnames like Sharma, Agrawal, Kumar etc
- <ne=NED>: Prefix matching with common designation like doctor, raja, pradhanmantri etc.

5.4 Voting

In Voting we use the results of CRF, MaxEnt and Rule Based model to get a better model. We have NE tags including "none". For each word the weight of these tags is initialized 0. Now when the word is predicted as some NE tag by a model then the weight of that tag is increased. The final answer is the tag which has highest weight.

Some heuristics are used to improve the accuracy of model. Like weight of NEM tags predicted by rule based model is kept high as they generally predict correct NE tag. If two tags are same then the answer is that tag.

6. DESIGN & IMPLEMENTATION

6.1 Data and Tools

- **Dataset:** Named Entity Annotated Corpus for Hindi. The data is obtained from IJCNLP-08 website [8]. SSF format [9] is used for representing the annotated Hindi corpus. The annotation was performed manually by IIIT Hyderabad.
- **Dictionary Source:** We have used files containing common Hindi nouns, verbs, adjectives, adverbs for Parts-of-speech (POS) tagging. The files are obtained from Hindi Wordnet, IIT Mumbai [11].
- **Tools:** Mallet-0.4 [12] is used for training and testing machine learning based models CRF [10] and MaxEnt [6]. For CRF, a SimpleTagger is provided which takes input as a file containing word followed by word features (noun, verb, number etc) and Named Entity (NE) tag for training. A SimpleTagger program converts the file into suitable data structures used by CRF for training.

  e.g. Training file format:

  ```
  Word feaure_1 feature_2 ... feature_n NE_tag
  ek noun adj number <ne=NEN>
  adhik adj adv none
  ```

  Here word "ek" has 3 features namely noun, adj and number. Its NE tag is <ne=NEN>. Second word "adhik" has 2 features namely adj and adv and it has NE tag none.

  For testing the file format is same except it doesn't contain NE tags at last of each sentence i.e. it only contains words followed by its features

  For MaxEnt, we created MaxEntTagger.java to process the input file and use them to test and train MaxEnt model.

- **Tagset Used:** Table 1 [2] contains the list Named Entity tagset used in the corpus.
- Programming Language & utility: Java, bash script, awk, grep

Tags	Names	Description
<ne=NEP>	Person	Bob Dylan, Mohandas Gandhi
<ne=NED>	Designation	General Manager, Commissioner
<ne=NEO>	Organization	Municipal Corporation
<ne=NEA>	Abbreviation	NLP, B.J.P.
<ne=NEB>	Brand	Pepsi, Nike (ambiguous)
<ne=NETP>	Title Person	Mahatma, Dr., Mr.
<ne=NETO>	Title Object	Pride and Prejudice, Othello
<ne=NEL>	Location	New Delhi, Paris
<ne=NETI>	Time	3rd September, 1991(ambiguous)
<ne=NEN>	Number	3.14, 4,500
<ne=NEM>	Measure	Rs. 4,500, 5 kg
<ne=NETE>	Terms	Maximum Entropy, Archeology
None	Not a named entity	Rain, go, hai, ka, ke , ki

TABLE 1: The named entity tagset used for shared task

6.2 Design Schemes

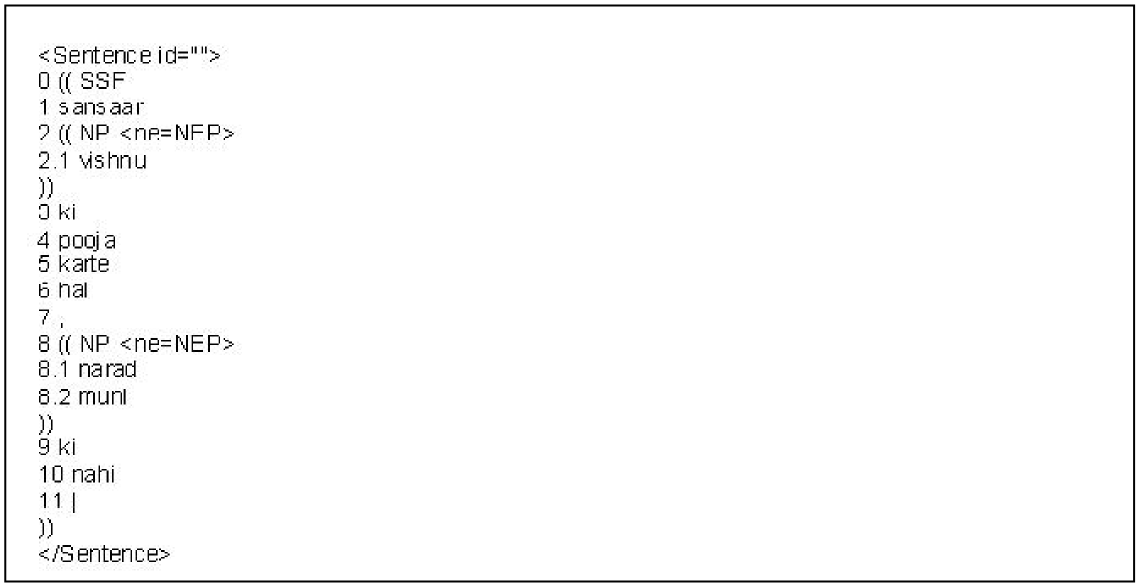
- **Editing Data:** The first objective is to convert annotated Hindi corpus given in SSF format to new format that can be used by mallet-0.4 models CRF and MaxEnt for training and testing. SSF format like the example given in Figure 2 contains many things like line number, braces, <Sentence id=""> etc that are not present in mallet format (e.g. data format of Figure 3). NE tags are present in different line in SSF, which need to put after the word for mallet format. Also some words which represents a NE tag when combined like "narad muni" in Figure 2 needs to be concatenated. After writing each word in different line with their NE tags, we need to find features for each word.

- **Features:** Here we used mostly orthographic features like other researchers have been using. Features of words include

    - Symbol: If the word is symbol like "?", ",", ";", "." etc
    - Noun: If word is noun
    - Adj: The word is adjective
    - Adv: adverb
    - Verb: verb
    - First Word: If the word is first word of a sentence
    - Number: If the word is a number like ek, paanch, or 123,
    - Num Start: If the word starts with number line 123_kg

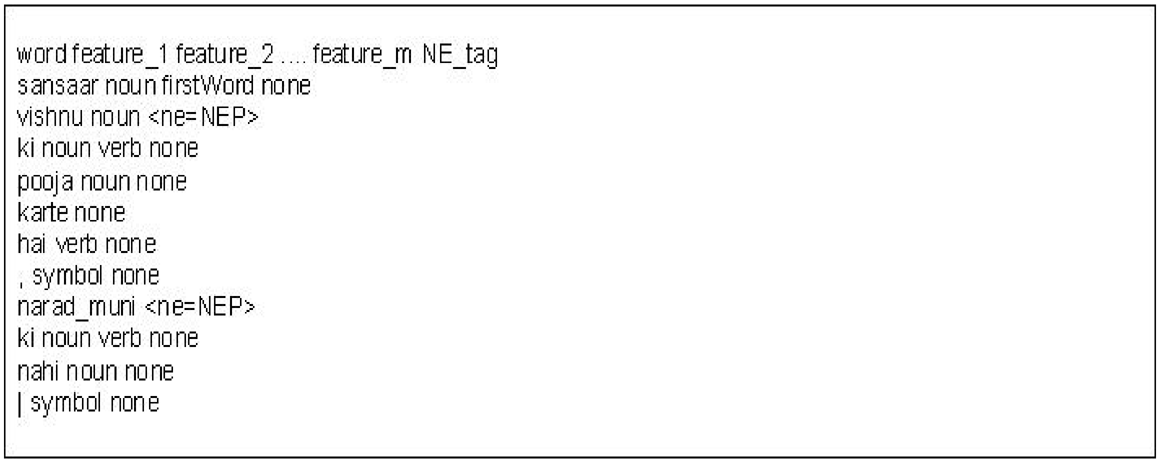
    Features of the words are added using some rule based matching (like for numbers) and from dictionary matching of words with the words which are obtained from Hindi wordnet, IIT Mumbai [11] (like noun, verb).

- **Training and Testing on Mallet:** The model is trained on 10, 50, 100, and 150 training files respectively. Then each trained model is tested on 10 files on which the model is not trained. The files on which the model is trained and tested are obtained randomly from the dataset. This process is done for 10 times. The average and good results of these tests are reported in the Results section. This is done for both CRF and MaxEnt model on the given data.



```
<Sentence id="">
0 (( SSF
1 sansaar
2 (( NP <ne=NEP>
2.1 vishnu
))
3 ki
4 pooja
5 karte
6 hai
7 ,
8 (( NP <ne=NEP>
8.1 narad
8.2 muni
))
9 ki
10 nahi
11 |
))
</Sentence>
```

FIGURE 2: Data in SSF format



```
word feature_1 feature_2 .... feature_m NE_tag
sansaar noun firstWord none
vishnu noun <ne=NEP>
ki noun verb none
pooja noun none
karte none
hai verb none
, symbol none
narad_muni <ne=NEP>
ki noun verb none
nahi noun none
| symbol none
```

FIGURE 3: Data in mallet format after conversion from SSF

- **Test Dataset using Rule Based Models:** test all datasets for Rule based models.
- **Improve Accuracy by Voting:** The output of each of the above method (CRF, MaxEnt, rule based) is file containing predicted tags for each word in the same line as the word. Voting algorithm uses trained CRF and MaxEnt model and rule based model's result and used the result of these to give better results. Voting is done on the results of these three models and the one with the most weight is the final tag.

7. RESULTS

7.1 Performance Evaluation Metric

The Evaluation measure for the data sets is precision, recall and F Measure.

- **Precision (P):** Precision is the fraction of the documents retrieved that are relevant to the user's information need.

$$\text{Precision(P)} = \frac{\text{correct answers}}{\text{answers produced}}$$

- **Recall (R):** Recall is the fraction of the documents that are relevant to the query that are successfully retrieved.

$$\text{Recall(R)} = \frac{\text{correct answers}}{\text{total possible correct answers}}$$

- **F-Measure:** The weighted harmonic mean of precision and recall, the traditional F-measure or balanced F-score is

$$F - Measure = \frac{(\beta^2 + 1)PR}{\beta^2 R + P}$$

β is the weighting between precision and recall typically $\beta = 1$.

When recall and precision are evenly weighted i.e. $\beta = 1$, F-measure is called F1-Measure.

$$F1 - Measure = \frac{2PR}{(P + R)}$$

There is a tradeoff between precision and recall in the performance metric.

7.2 Results Obtained

- **CRF Results:** The following table contains the results obtained from testing CRF models. The model is trained on 10, 50, 100 and 150 files and then tested on 10 files. This is done for 10 rounds i.e. for model trained on 100 files, 110 files are selected from the dataset and it is trained on 100 files and tested on 10 files(model trained on 10 files are tested on 5 files). Then again 110 files are chosen and training and testing is done. This is done for 10 times. Table 2 contains the results obtained from the above experiment.

Number of training files	Number of testing files	Precision	Recall	F-1 Measure
10	5	71.43	30.86	43.10
50	10	83.87	25.74	39.40
100	10	88.24	24.19	37.97
150	10	88.89	24.61	38.55

TABLE 2: CRF results for one best predicted tag

For the above experiments only one predicted tag of a word is considered. Since the number of NE tags are less compared to “none” tag, so the model learns mostly for “none” tag. So we considered using best of two of the predicted tags of a word to check the results. Here two best predicted tags are given by the model. The two tags can be either same or different. If first tag is a NE tag then that tag is considered correct. If first is none tag and second is NE tag then second tag is considered for the results. This experiment is also conducted in a similar manner as the above experiment.

The results obtained from the above experiment for CRF when two of the best predicted tags are taken into consideration is shown in the Table 3:

Number of training files	Number of testing files	Precision	Recall	F-1 Measure
10	5	70.0	34.57	46.28
50	10	89.28	49.5	63.69
100	10	83.33	33.9	48.19
150	10	74.28	33.37	46.43

TABLE 3: CRF results for best of two predicted tags

- **MaxEnt Results:** Following tables contain the results of training and testing of MaxEnt model. The model is trained on randomly chosen 10, 50, 100 and 150 files and then tested on 10 files on which it is not trained. Each of the training and testing is done for ten rounds. Similar to above these are also tested on different datasets. The results obtained is shown in the following table 4:

Number of training files	Number of testing files	Precision	Recall	F-1 Measure
10	5	76.92	19.8	31.49
50	10	70.40	16.68	26.39
100	10	69.21	18.14	28.19
150	10	69.46	16.57	26.06

TABLE 4: MaxEnt Results for one best predicted tag

MaxEnt results when two of the best predicted tags are taken into consideration are given in Table 5. This is done in similar way as done in CRF experiment.

Number of training files	Number of testing files	Precision	Recall	F-1 Measure
10	5	90.47	29.23	44.18
50	10	89.28	21.36	34.48
100	10	87.5	22.58	35.89
150	10	96.15	25.25	39.99

T

TABLE 5: MaxEnt Results for best of two predicted tags

- **Rule Based Results:** Results driven from rule based model is given below in Table 6:

Number of testing files	Precision	Recall	F-1 Measure
1	65.93	77.92	71.43
2	88.0	60.27	71.54
3	96.05	86.90	91.25

TABLE 6: Rule based model's test results

- **Voting Algorithm:** For voting we used three classifiers crf trained on 50 files, MaxEnt trained on 50 files and rule based. Results from voting algorithm model is given in Table 7:

Number of testing files	Precision	Recall	F-1 measure
40	81.11	84.88	82.95
40	85.51	76.62	80.82

TABLE 7: Voting Algorithm's Results

8. CONCLUSION

Basically this paper presents a comparative study among different approaches like MaxEnt, CRF and Rulebase using POS & orthographic features. It also shows that voting mechanism gives the better results. On average CRF gives better result than MaxEnt. Rule based result has better recall and F-1 measure. On the given data the average precision is good. The main reason for the lower F-1 measure by CRF and MaxEnt is due to the presence of less NE tags in the original data compared to "none". For most file the percentage of NE tags is less that 2% of the total words present in a file. Because of that the classifier is learned more strongly for "none" rather than NE tags. Also data has tagging errors. e.g. "Gandhi" is classified as <ne=NEN>, <ne=NEP>, <ne=NED>,"none" in many files. Similarly "ek" is classified as <ne=NEN> or "none". These conflicting cases in the training set weaken the classifier. That's why more training doesn't give better results here. The classifier gives good precisions i.e. less tags are classified but they are classified correctly.

When we took best of two predicted tags for the results analysis F-1 measure and recall increases significantly. Since we have very few NE tags in data and also data is not very accurate, so most of the words are learned as "none", but when we consider best of two predicted tags, the result improves significantly. Rule based model gives better average result (F-1 measure, recall) for given data. Voting algorithm improves the F-1 measure of results.

9. FUTURE WORK

Dictionary matching of words is not very effective. In this experiment we used Othographic features like other researchers however POS tagger or morphological analyzer, semantic tags, parasargs (prepositions and postpositions) identification, lexicon database and co-occurrences may give the better results. Boosting may be done by containing 5 words above NE tags and 5 words below NE tags. Conflicting tags can be removed. Or we may try using another dataset. More features can be added to improve the models. Rule based model can be improved. We may experiment with other classifier like HMM.

10. ACKNOWLEDGMENT

I would like to thank Mr. Pankaj Srivastava, Ms. Agrima Srivastava and MS. Vertika Khanna who provide helpful analysis in model development.

11. REFERENCES:

[1] Sudeshna Sarkar, Sujan Saha and Prthasarthi Ghosh, "Named Entity Recognition for Hindi", In Microsoft Research India Summer School talk, p. 21-30, May 2007.

[2] Anil Kumar Singh, "Named Entity Recognition for South and South East Asian Languages: Taking Stock", p. 5-7, In IJCNLP 2008.

[3] Hideki Isozaki. 2001. “Japanese named entity recognition based on a simple rule generator and decision tree learning” in the proceedings of the Association for Computational Linguistics, pages 306-313. India.

[4] Takeuchi K. and Collier N. 2002. “Use of Support Vector Machines in extended named entity recognition” in the proceedings of the sixth Conference on Natural Language Learning (CoNLL-2002), Taipei, Taiwan, China.

[5] Charles L. Wayne. 1991., “A snapshot of two DARPA speech and Natural Language Programs” in the proceedings of workshop on Speech and Natural Languages, pages 103-404, Pacific Grove, California. Association for Computational Linguistics.

[6] A. Borthwick, "A Maximum Entropy Approach to Named Entity Recognition", In NY University, p. 1-4, 18-24, PHD Thesis, September 1999

[7] Daniel M. Bikel, Scott Miller, Richard Schwartz and Ralph Weischedel. 1997 “Nymble: a high performance learning name-finder” in the proceedings of the fifth conference on Applied natural language processing, pages 194-201, San Francisco, CA, USA Morgan Kaufmann Publishers Inc.

[8] IJCNLP-08 Workshop data set, Source: http://ltrc.iiit.net/ner-ssea-08/index.cgi?topic=5

[9] Akshar Bharti, Rajeev Sangal and Dipti M Sharma, "Shakti Analyzer: SSF Representation", IIIT Hyderabad, p. 3-5, 2006

[10] Lafferty, J., McCallum, A., Pereira, F., "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", In: Proc. 18th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, p. 1-5, 2001

[11] Hindi Wordnet, Source: http://www.cfilt.iitb.ac.in/wordnet/webhwn/

[12] McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu. 2002.

[13] Hanna M. Wallach, "Conditional Random Fields: An Introduction”, Technical Report, University of Pennsylvania. 4-5, 2004.

[14] Lawrence R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", In Proceedings of the IEEE, 77 (2), p. 257-286,February 1989

[15] R. Grishman. 1995. “The NYU system for MUC-6 or Where’s the Syntax” in the proceedings of Sixth Message Understanding Conference (MUC-6) , pages 167-195, Fairfax, Virginia.

[16] Wakao T., Gaizauskas R. and Wilks Y. 1996. “Evaluation of an algorithm for the Recognition and Classification of Proper Names”, in the proceedings of COLING-96.

[17] Mikheev A, Grover C. and Moens M. 1998. Description of the LTG system used for MUC-7. In Proceedings of the Seventh Message Understanding Conference.

[18] R. Grishman, Beth Sundheim. 1996. “Message Understanding Conference-6: A Brief History” in the proceedings of the 16th International Conference on Computational Linguistics (COLING), pages 466-471, Center for Sprogteknologi, Copenhagen, Denmark.

[19] Srihari R., Niu C. and Li W. 2000. A Hybrid Approach for Named Entity and Sub-Type Tagging. In: Proceedings of the sixth conference on applied natural language processing.

[20] Cucerzan S. and Yarowsky D. 1999. Language independent named entity recognition combining morphological and contextual evidence. In: Proceedings of the Joint SIGDAT Conference on EMNLP and VLC 1999, pp. 90-99.

[21] Li W. and McCallum A. 2003. Rapid Development of Hindi Named Entity Recognition using Conditional Random Fields and Feature Induction. In: ACM Transactions on Asian Language Information Processing (TALIP), 2(3): 290–294.

[22] Gali, K., Sharma, H., Vaidya, A., Shisthla, P., Sharma, D.M.: Aggregrating Machine Learning and Rule-based Heuristics for Named Entity Recognition. In: Proceedings of the IJCNLP-08Workshop on NER for South and South East Asian Languages. (2008) 25–32

[23] Asif Ekbal et. al. “Language Independent Named Entity Recognition in Indian Languages”. IJCNLP, 2008.

[24] Prasad Pingli et al. “A Hybrid Approach for Named Entity Recognition in Indian Languages”. IJCNLP, 2008.

[25] Shilpi Srivastava, Siby Abraham, Mukund Sanglikar: “Hybrid Approach for Recognizing Hindi Named Entity”, Proceedings of the International Conference on Managing Next Generation Software Applications - 2008 (MNGSA 2008), Coimbatore, India, 5th- 6th December 2008.

[26] Shilpi Srivastava, Siby Abraham, Mukund Sanglikar, D C Kothari: “Role of Ensemble Learning in Identifying Hindi Names”, International Journal of Computer Science and Applications, ISSN No. 0974-0767.

Language Combinatorics: A Sentence Pattern Extraction Architecture Based on Combinatorial Explosion

Michal Ptaszynski *ptaszynski@hgu.jp*
High-Tech Research Center
Hokkai-Gakuen University
Sapporo, 064-0926, Japan

Rafal Rzepka *kabura@media.eng.hokudai.ac.jp*
Graduate School of Information Science and Technology
Hokkaido University
Sapporo, 060-0814, Japan

Kenji Araki *araki@media.eng.hokudai.ac.jp*
Graduate School of Information Science and Technology
Hokkaido University
Sapporo, 060-0814, Japan

Yoshio Momouchi *momouchi@eli.hokkai-s-u.ac.jp*
Department of Electronics and Information Engineering,
Faculty of Engineering
Hokkai-Gakuen University
Sapporo, 064-0926, Japan

Abstract

A “sentence pattern” in modern Natural Language Processing is often considered as a subsequent string of words (n-grams). However, in many branches of linguistics, like Pragmatics or Corpus Linguistics, it has been noticed that simple n-gram patterns are not sufficient to reveal the whole sophistication of grammar patterns. We present a language independent architecture for extracting from sentences more sophisticated patterns than n-grams. In this architecture a “sentence pattern” is considered as n-element ordered combination of sentence elements. Experiments showed that the method extracts significantly more frequent patterns than the usual n-gram approach.

Keywords: Pattern Extraction, Corpus Pragmatics, Combinatorial Explosion.

1. INTRODUCTION

Automated text analysis and classification is a typical task in Natural Language Processing (NLP). Some of the approaches to text (or document) classification include Bag-of-Words (BOW) or n-gram. In the BOW model, a text or document is perceived as an unordered set of words. BOW thus disregards grammar and word order. An approach in which word order is retained is called the n-gram approach, proposed by Shannon over half a century ago [22]. This approach perceives a given sentence as a set of n-long ordered sub-sequences of words. This allows for matching the words while retaining the sentence word order. However, the n-gram approach allows only for a simple sequence matching, while disregarding the grammar structure of the sentence. Although instead of words one could represent a sentence in parts of speech (POS), or dependency structure, the n-gram approach still does not allow for extraction or matching of more sophisticated patterns than the subsequent strings of elements. An example of such pattern, more sophisticated than n-gram, is presented in top part of Figure 1. A sentence in Japanese “*Kyō wa nante kimochi ii hi nanda !*” (What a pleasant day it is today!) contains a syntactic pattern "*nante * nanda !*"[1]. Similar cases can be easily found in other languages, for instance, English and Spanish. An exclamative sentence "Oh, she is so pretty, isn't she?", contains a syntactic pattern "Oh * is so * isn't *?". In Columbian Spanish, sentences "*¡Qué majo está carro!*" (What a nice car!) and "*¡Qué majo está chica!*" (What a nice girl!) contain a common pattern "*¡Qué majo está * !*" (What a nice * !). With another sentence, like "*¡Qué porquería de película!*" (What a crappy movie!) we can obtain a higher level generalization of this pattern, namely "*¡Qué * !*" (What a * !), which is a typical *wh*-exclamative sentence pattern [15]. The existence of such patterns in language is common and well recognized. However, it is not possible to discover such subtle patterns using only n-gram approach. Methods trying to go around this problem include a set of machine learning (ML) techniques, such as Neural Networks (NN) or Support Vector Machines (SVM). Machine learning has proved its usefulness for NLP in text classification within different domains [21, 16]. However, there are several problems with the ML approach. Firstly, since machine learning is a self-organizing method, it disregards any linguistic analysis of data, which often makes detailed error analysis difficult. Moreover, the statistical analysis performed within ML is still based on words (although represented as vectors), which hinders dealing with word inflection and more sophisticated patterns such as the ones mentioned above. Although there are attempts to deal with this problem, like the string kernel method [12], in ML one always needs to know the initial training set of features to feed the algorithm. Finally, methods for text classification are usually inapplicable in other tasks, such as language understanding and generation.

In our research we aimed to create an architecture capable to deal or help dealing with the above problems. The system presented in this paper, SPEC, extracts from sentences patterns more sophisticated than n-grams, while preserving the word order. SPEC can work with one or more corpora written in any language. The corpora can be raw or preprocessed (spaced, POS tagging, etc.). This way SPEC extracts all frequent meaningful linguistic patterns from unrestricted text. This paper presents general description of SPEC, evaluates several aspects of the system performance and discusses possible applications.

The paper outline is as follows. In section 2 we present background and motivation for the research, and explain general terms frequently used in this paper. Section 3 contains detailed description of all system procedures and modules put to the evaluation. In section 4 we describe the experiments performed to evaluate the system in several aspects influential for its performance. Finally, we discuss the results in section 5 and conclude the paper in section 6.

2. BACKGROUND

2.1 Corpus Pragmatics

Pragmatics is a subfield of linguistics focusing on the ways natural language is used in practice [11]. In general it studies how context of a sentence influences its meaning. There are, roughly, two approaches to this problem. Classic approach is to look for "hidden meanings" of a sentence, called implicatures [6]. Another approach takes as an object a corpus (a coherent collection of texts) and analyzes examples of certain language strategies within their contexts to study their functions. For this it has been named Corpus Pragmatics (differently to Corpus Linguistics, which does not put so much focus on context, but rather on the word examples alone). Some of the research in Corpus Pragmatics has been done by Knight and Adolphs [7], Potts and Schwarz [15], or Constant, Davis, Potts and Schwarz [4]. Especially the latter two have focused on emotive utterances. They have used a corpus of reviews from Amazon.com, and analyzed tokens (words, usually unigrams to trigrams) that were the most distinguishable for emotively emphasized reviews (marked very low or very high). The main drawback of these research was focusing only on words, while disregarding both grammatical information, like POS or dependency structure, and more sophisticated patterns, like the ones mentioned in Introduction. With this paper we wish to contribute to the field of Corpus Pragmatics by providing an architecture capable of automatic

[1] equivalent of *wh*-exclamatives in English [20]; asterisk used as a wildcard.

extraction of such patterns from corpora. In our assumption this could be done by generating all patterns as ordered combinations of sentence elements and verifying the occurrences of all patterns within a corpus. This introduces to our research a problem of Combinatorial Explosion.

2.2 Combinatorial Explosion

Algorithms using combinatorial approach generate a massive number of combinations - potential answers to a given problem. This is the reason they are sometimes called brute-force search algorithms. Brute-force approach often faces the problem of exponential and rapid grow of the function values during combinatorial manipulations. This phenomenon is known as combinatorial explosion [9]. Since this phenomenon often results in very long processing time, combinatorial approaches have been often disregarded. We assumed however, that combinatorial explosion can be dealt with on modern hardware to the extent needed in our research. Moreover, optimizing the combinatorial approach algorithm to the problem requirements should shorten the processing time making combinatorial explosion an advantage in the task of pattern extraction from sentences.

2.3 Pattern Extraction

Pattern extraction from language corpora is a subfield of Information Extraction (IE). There is a number of research dealing with this task or applying pattern extraction methods to solve other problems. Some of the research related the most to ours include Riloff 1996 [18], Uchino et al. 1996 [25], Talukdar et al. 2006 [24], Pantel and Pennacchiotti 2006 [14] or Guthrie et al. [23]. Riloff [18] proposed AutoSlog-TS system, which automatically generates extraction patterns from corpora. However, their system, being in fact an updated version of previous AutoSlog, was created using manually annotated corpus and a set of heuristic rules. Therefore the system as a whole was not fully automatic. Moreover, patterns in their approach were still only n-grams. A similar research was reported by Uchino et al. [25]. They used basic phrase templates to automatically expand the number of template patterns and applied them to machine translation. They also focused only on n-gram based patterns. Research tackling patterns more sophisticated than n-grams was done by Talukdar et al. [24]. They proposed a context pattern induction method for entity extraction. However, in their research the seed word set was provided manually and the extraction limited to the patterns neighboring the seed words. Therefore the patterns in their research were limited to n-grams separated with one word inside the pattern. Moreover, their system disregarded grammatical information. Espresso, a system using grammatical information in pattern extraction was reported by Pantel and Pennacchotti [14]. Espresso used generic patterns to automatically obtain semantic relations between entities. However, although the patterns Espresso used were not limited to n-grams, they were very generic (e.g. is-a or part-of patterns) and were provided to the system manually. An idea close to ours, called “skipgrams” was proposed Guthrie et al. [23]. The idea assumed that “skips” could be put between elements of n-grams (similar to wildcards in SPEC). However, they focused only on bigrams and trigrams. Moreover, they assumed that n-gram elements could be separated at most by 4 skips, which makes the extraction of patterns shown in Introduction impossible. In comparison with the mentioned methods, our method is advantageous in several ways. Firstly, we aimed to fully automatize the process of generation of potential patterns and extraction of actual patterns. Secondly, we deal with patterns more sophisticated than n-grams, generic separated patterns or skipgrams.

3. SPEC - SYSTEM DESCRIPTION

This section contains detailed description of SPEC, or *Sentence Pattern Extraction arChitecturte*. In the sections below we describe the system sub-procedures. This includes corpus preprocessing, generation of all possible patterns, extraction of frequent patterns and post-processing. By a “corpus” we consider any collection of sentences or instances. It can be very large, containing thousands of sentences, or small consisting of only several or several dozen sentences. In any case SPEC automatically extracts frequent sentence patterns distinguishable for the corpus. In the assumption, the larger and the more coherent the original corpus is, the more frequent patterns will be extracted.

3.1 Corpus Preprocessing

SPEC was designed to deal with any not preprocessed raw corpora, as long as the lexical form of the language consists of smaller distinguishable parts, like letters, or characters. This makes SPEC capable to deal with corpora written in any type of language, including analytic languages (like English or Mandarin Chinese), agglutinative languages (like Japanese, Korean or Turkish), or even polysynthetic languages like Ainu, in their both spaced and non-spaced form. However, in the Pattern Generation sub-procedure, SPEC creates a very large number of temporary patterns (all possible ordered combinations of sentence elements). Therefore, considering the processing time, to avoid extensive combinatorial explosion, as a default we will assume that the corpus is at least spaced. Other relevant optional preprocessing might include part-of-speech (POS) tagging, dependency relation tagging or any other additional information as long as there exist sufficient tools. Three examples of preprocessing with and without POS tagging are presented in Table 1 for a sentence in Japanese[2]. The sentence in the example was spaced and tagged with MeCab [10], a standard POS tagger for Japanese.

Sentence:	今日はなんて気持ちいい日なんだ！
Transliteration:	*Kyōwanantekimochiiihinanda!*
Meaning:	Today TOP what pleasant day COP EXCL
Translation:	What a pleasant day it is today!
	Preprocessing examples
1. Words:	*Kyō wa nante kimochi ii hi nanda !*
2. POS:	N TOP ADV N ADJ N COP EXCL
3.Words+POS:	*Kyō*[N] *wa*[TOP] *nante*[ADV] *kimochi*[N] *ii*[ADJ] *hi*[N] *nanda*[COP] *!*[EXCL]

TABLE 1: Three examples of preprocessing of a sentence in Japanese with and without POS tagging; N = noun, TOP = topic marker, ADV = adverbial particle, ADJ = adjective, COP = copula, INT = interjection, EXCL = exclamative mark.

3.2 Pattern Generation

In this procedure SPEC generates all possible combinations of patterns from a sentence. Various algorithms have been proposed for creating combinations. Some use iteration loops, other use recursion. As processing speed is a crucial factor when dealing with reasonable size corpora, we designed two versions of a module to perform this procedure. The first version was designed to use one of the officially available iteration based algorithms for combination generation. In the second version we used a recursion algorithm designed especially for the task. Below we describe both versions. In section 4 we perform a speed test on four different iterative algorithms to choose the fastest one. The version of SPEC based on the fastest iterative algorithm is confronted later with the recursive version.

Iteration Based Algorithm

Generation of All Combinations from Sentence Elements

In this sub-procedure, the system generates ordered non-repeated combinations from the elements of the sentence. In every *n*-element sentence there is *k*-number of combination groups, such as $1 \leq k \leq n$, where *k* represents all *k*-element combinations being a subset of *n*. The number of combinations generated for one *k*-element group of combinations is equal to binomial coefficient, represented in equation 1. In this procedure we create all combinations for all values of *k* from the range of *{1,…,n}*. Therefore the number of all combinations is equal to the sum of all combinations from all *k*-element groups of combinations, like in the equation 2.

[2] Japanese is a non-spaced agglutinative language.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

$$\sum_{k=1}^{n} \binom{n}{k} = \frac{n!}{1!(n-1)!} + \frac{n!}{2!(n-2)!} + \dots + \frac{n!}{n!(n-n)!} = 2^n - 1 \tag{2}$$

Ordering of Combinations
In mathematics, combinations are groups of unordered elements. Therefore, using only the above algorithm, we would obtain patterns with randomized order of sentence elements, which would make further sentence querying impossible. To avoid randomization of sentence elements and preserve the sentence order we needed to sort each time the elements of a combination after the combination has been generated. To do that we used automatically generated double hash maps. Firstly, all elements of the original input sentence are assigned ordered numbers (1, 2, 3...). After a combination is generated, elements of this combination are re-assigned numbers corresponding to the numbers assigned to the original sentence elements. The new set of numbers is sorted. Then the sorted list of numbers is re-mapped on original sentence elements (words) using the first hash. This provides the appropriate order of combination elements consistent with the order of elements in the original sentence. See Figure 1 for details of this part of the procedure.

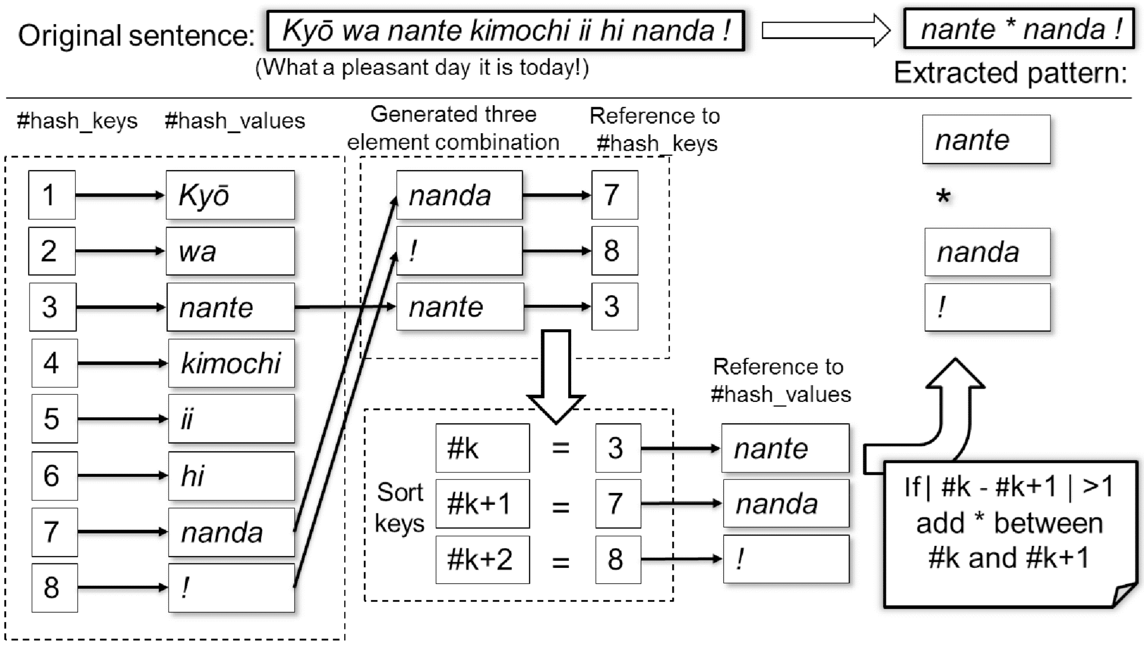


FIGURE 1: The procedure for sorting combination elements with automatically generated hash maps.

Insertion of Wildcard
In this stage the elements of a pattern are already sorted, however, to perform effective queries to a corpus we would also need to specify whether the elements appear next to each other or are separated by a distance. In practice, we need to place a wildcard between all non-subsequent elements. We solved this using one simple heuristic rule. If absolute difference of hash keys assigned to the two subsequent elements of a combination is higher than 1, we add a wildcard between them. This way we obtain a set of ordered combinations of sentence elements with wildcards placed between non subsequent elements. All parts of this procedure: generation of

combinations, sorting of elements using automatically generated hash maps and wildcard insertion, are represented in Figure 1.

Recursion Based Algorithm

The recursion based algorithm is a coroutine-type recursive generator, which produces a new item in a list each time it is called. It returns all the elements in the list passed into it, unless the previous element is the same (this removes adjacent wildcards). The algorithm flow goes as follows. Firstly, it writes out all possible combinations of the word list by continuously replacing subsequent words with wildcards. This operation is recursively called by itself, beginning from 0 wildcards to the overall number of sentence elements. The algorithm goes through all positions in the word list (sentence) starting from the beginning and places a wildcard there. This prevents infinite loops, since, if it goes beyond the end of the list, it will fall through without executing the loop. The original value is saved off and a wildcard is placed at this position. Then it calls itself recursively on the next index and with one less wildcard left to place. The list is restored to its original position and the operation is repeated till there is no more wildcards to place. Finally adjacent wildcards are removed and output is written to files. The whole procedure is performed for all sentences in the corpus.

3.3 Pattern Extraction and Pattern Statistics Calculation

In this sub-procedure SPEC uses all original patterns generated in the previous procedure to extract frequent patterns appearing in a given corpus and calculates their statistics. The statistics calculated include *number of pattern occurrences (O)*, *pattern occurrence frequency (PF)* and *pattern weight (W)*.

Number of pattern occurrences (O) represents the number of all occurrences of a certain *k*-long pattern in a given corpus.

Pattern occurrence frequency (PF) represents the number of all occurrences of a *k*-long pattern within a corpus divided by the number of all *k*-long patterns that appeared more than once *(Ak)*. See formula 3.

Pattern weight (W) is a multiplication of the length of *k* and *PF*. See formula 4.

$$PF_k = \frac{O}{A_k} \qquad (3)$$

$$W_k = k * PF_k \qquad (4)$$

The general pragmatic rule which applies here says that the longer the pattern is (length *k*), and the more often it appears in the corpus (occurrence *O*), the more specific and representative it is for the corpus (weight *W*). The pattern frequency calculation can be performed as a part of the previous procedure (pattern generation) when dealing with only one corpus. However, an often need in tasks like document classification is to obtain a set of patterns from a training (often smaller) corpus and perform pattern matching on a larger corpus. For example, in lexicon expansion one uses seed patterns to find out new vocabulary and phrases appearing within the patterns. We assumed SPEC should be able to deal with both, single and multiple corpus case. Therefore we needed to retain the ability to generate patterns from one (given) corpus and match them to another (target) corpus. Separating pattern generation and pattern extraction procedures also allows cross-reference of two or more corpora (a case when both are given and target corpora). This also allows performing the extraction on all corpora concurrently, e.g., using *fork* or *thread* functions for parallel programming, which shortens processing time. Finally, by making the system module-based (all sub-procedures are created as separate modules), each sub-procedure can be thoroughly evaluated, improved, expanded, or substituted with more efficient one when needed.

3.4 Post Processing

In the post-processing phase SPEC performs simple analysis of patterns extracted from the given corpus to provide its basic pragmatic specifications. We use the term "pragmatic specifications" in a similar way to Burkhanov, who generally includes here indications and examples of usage [2]. The post-processing is done differently for: 1) one given/target corpus and 2) a set of two or more corpora.

One Corpus Case

The post-processing of one corpus is done as follows. Firstly, all patterns that appeared only once are filtered out and deleted. This is done to eliminate quasi-patterns. A quasi-pattern is a pattern, which was created from a sentence in the combinatorial explosion-based process of pattern generation, but was not found elsewhere in the rest of the corpus. In practice, it means that it is not a frequently used sentence pattern and therefore keeping it would bias the results. The patterns that appeared more than once are grouped according to pattern length (number of elements a pattern consists of). The patterns are also indexed within groups using initial pattern letters as indices. In this form the patterns can be used in further analysis.

Two/Multiple Corpora Case

In many NLP tasks, especially those taking advantage of machine learning methods, it is often necessary to obtain lists of two distinctive sets of features [5]. This refers to all sorts of text classification domains, like spam filtering, sentiment and affect analysis [13] or even novel ones, like cyber-bullying detection [16]. In the post-processing procedure SPEC refines the patterns extracted for several corpora to filter out the ones that appear exclusively in each corpus and the ones which occurrences repeat in several corpora. The post-processing for two corpora is done as follows. Firstly, SPEC deletes only those quasi-patterns that appeared only once in both corpora. For all patterns which appeared more than once in at least one corpus SPEC calculates for which of the two corpora they are more representative, applying the notions of *O*, *PF* and *W* defined in section **3.3**. Overall diagram of the SPEC system is represented in Figure 2.

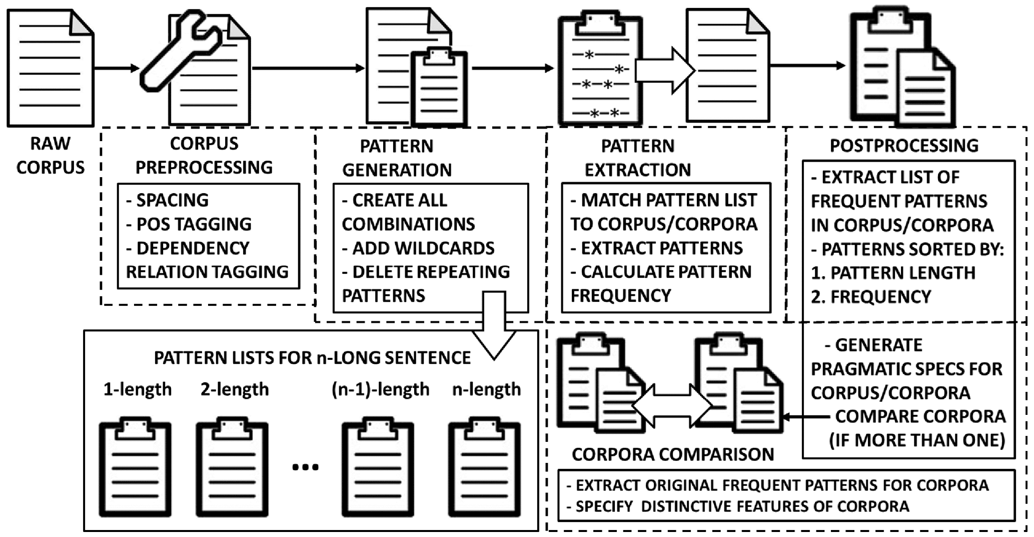

RAW CORPUS
CORPUS PREPROCESSING
- SPACING
- POS TAGGING
- DEPENDENCY RELATION TAGGING
PATTERN GENERATION
- CREATE ALL COMBINATIONS
- ADD WILDCARDS
- DELETE REPEATING PATTERNS
PATTERN EXTRACTION
- MATCH PATTERN LIST TO CORPUS/CORPORA
- EXTRACT PATTERNS
- CALCULATE PATTERN FREQUENCY
POSTPROCESSING
- EXTRACT LIST OF FREQUENT PATTERNS IN CORPUS/CORPORA
- PATTERNS SORTED BY:
1. PATTERN LENGTH
2. FREQUENCY
- GENERATE PRAGMATIC SPECS FOR CORPUS/CORPORA
COMPARE CORPORA (IF MORE THAN ONE)
PATTERN LISTS FOR n-LONG SENTENCE
1-length
2-length
(n-1)-length
n-length
...
CORPORA COMPARISON
- EXTRACT ORIGINAL FREQUENT PATTERNS FOR CORPORA
- SPECIFY DISTINCTIVE FEATURES OF CORPORA

FIGURE 2: General diagram of the SPEC system.

4. EXPERIMENTS

This section describes experiments we performed to evaluate different aspects of SPEC. As time is a crucial issue in systems using combinatorial approach we begin with experiments on processing speed. Next, we perform a detailed analysis of patterns extracted by SPEC with comparison to a usual n-gram approach.

4.1 Test Sets

The experiments are performed on two different test sets. Firstly, as was shown by Potts et al. [15] and Ptaszynski et al. [17], pragmatic differences are the most visible in a comparison of emotional and neutral language. Therefore, we used a set of emotive and non-emotive utterances in Japanese collected by the latter. There are 38 emotive utterances and 39 non-emotive utterances in the set.

4.2 Experiments with Processing Time

All experiments were conducted on a PC with the following specifications. Processor: Intel Core i7 980X; Memory: 24 GB RAM; Storage: SSD 256 GB; OS: Linux Fedora 14 64bit.

Speed Test of Iterative Algorithms for Combination Generation

In the first experiment we compared speed in generating combinations for four officially available algorithms. As most of SPEC procedures are written in Perl, we used algorithms designed for this programming language.

Math::Combinatorics (M::C) is a Perl module designed by Allen Day. It provides a pure-perl implementation of functions like combination, permutation, factorial, etc. in both functional and object-oriented form. The module is available at: http://search.cpan.org/~allenday/Math-Combinatorics-0.09/

Algorithm::Combinatorics (A::C) is also a Perl module for generation of combinatorial sequences, designed by Xavier Noria. The algorithms used in this module were based on professional literature [8]. The module is available at: http://search.cpan.org/~fxn/Algorithm-Combinatorics/

Algorithm α is a subroutine based iteration algorithm keeping track of a list of indices.

Algorithm β is also a subroutine based iteration algorithm, although optimized. Both α and β algorithms are available on PerlMonks, a website for Perl programming community, at: http://www.perlmonks.org/?node id=371228

The above four algorithms were applied in a simple task of generating all combinations from a list containing all letters of alphabet[3]. Processing time was calculated separately for each *k*-long group of combinations. The test was taken ten times for each group. This provided 260 tests for each algorithm, giving over a thousand of overall number of tests. For each group of tests the highest and lowest outliers were excluded and the averages of processing times were calculated. The results for all four tested algorithms are represented in Figure 3.

Iteration Algorithm vs. Recursion Algorithm

In the second experiment we compared two versions of pattern generation procedure, based on the fastest iteration algorithm and recursion algorithm described in section **3.2**. The task was to generate from the same list (alphabet letters) not only combinations, but the whole patterns, with sorted elements and wildcards included. Here the time was calculated not for each combination group but for the whole process of generating all patterns. The test was also taken ten times and the highest and lowest outliers excluded. The averages of results are represented in table 2.

[3] 26 letters: "a b c d e f g h i j k l m n o q p r s t u w v x y z".

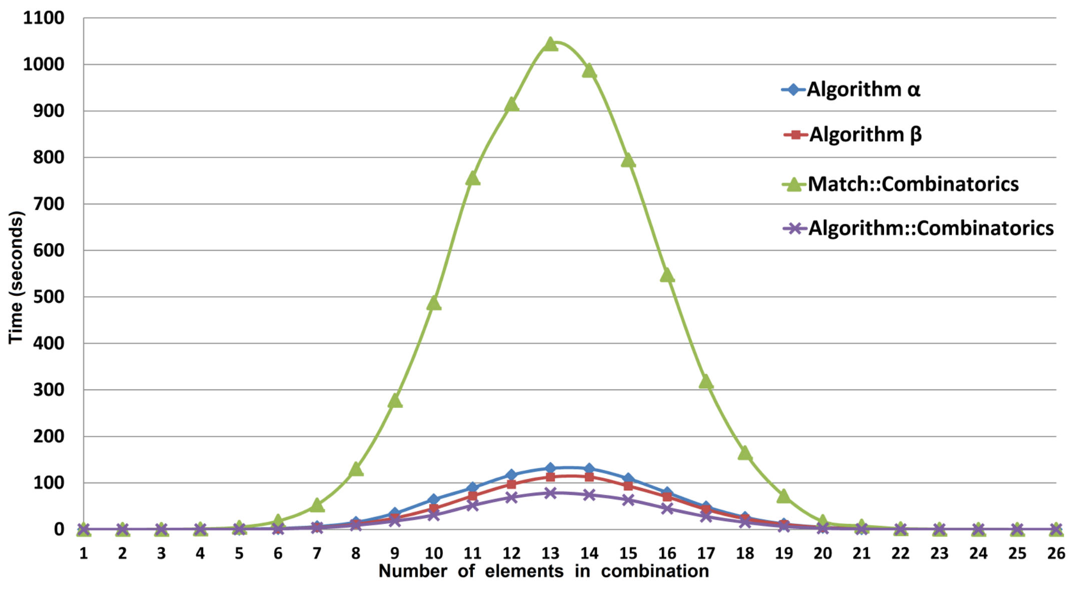


FIGURE 3: Graph showing time scores of all four tested algorithms.

Algorithm	Processing times (min., sec.)
Recursion based	24 min., 50.801 sec.
Iteration based	45 min., 56.125 sec.

TABLE 2: Averaged results of processing time compared between recursive and iterative version of pattern generation procedure.

4.3 Experiments with Pattern Analysis

N-gram Approach vs. Pattern Approach

By extracting frequent sentence patterns from corpora, SPEC builds a sentence pattern based language model from scratch for any corpus. An approach usually applied in language modeling is n-gram approach [1, 3, 19]. Therefore, for a comparison we used n-gram approach. In the experiment we calculated the number of extracted frequent patterns and compared it to the number of frequent n-grams. However, n-grams are also types of patterns and SPEC would generate n-grams as well. Therefore in this experiment we compared the number of extracted n-grams with the number of only non-n-gram patterns. As we assumed, this should show how effective is the pattern based method over n-grams. It might seem that the pattern based method would always perform better, since it generates incomparably larger numbers of patterns. However, in the evaluation we used only those patterns, which appeared in corpus more than once. This means we filter out all potential (or quasi) patterns leaving only the frequent ones. Thus the actual differences might not be of that high order. We also verified whether the differences between the numbers of frequent patterns and frequent n-grams were statistically significant using Student's T-test. Since the compared groups are of the same type (*k*-long patterns) we decided to use paired version of T-test. If the differences were small and not significant, it would mean that the traditional n-gram approach is equally good while much faster than the proposed pattern approach. We also compared the patterns in two different ways. Quantitative, referring to the overall number of patterns per group and Qualitative, in which we also compared how frequent were the patterns within a group. This way, a group with smaller number of very frequent patterns, could score higher than a group of many modestly frequent patterns.

5. RESULTS AND DISCUSSION

In the processing speed experiment, in which we compared four combinatorial algorithms, the fastest score was achieved by Algorithm::Combinatorics. The worst was Math::Combinatorics. Other two algorithms achieved similar results. However, none of them was faster than A::C. Therefore we used this algorithm in iterative version of SPEC.

Next, we compared the processing time of pattern generating procedure for iteration and recursion based algorithms. The latter one was over two times faster (see Table 2). Although the iterative algorithm itself is fast, additional operations performed after generating the combinations, such as sorting of elements and insertion of wildcards, influence the overall processing time. Therefore for the task of generating all combinations, it is more efficient to use the recursion based algorithm. The further analysis of patterns, however, revealed that it is not always necessary to create all patterns. The experiment showed that generating up to 5-element combinations is sufficient to find all frequent language patterns from a corpus (see Figure 4). This discovery, when confirmed on larger datasets, on corpora of different languages and domains, should provide some deep insights about the nature of structured language patterns, like compound nouns, collocations or even conversational strategies, which would contribute greatly to the field of Corpus Pragmatics.

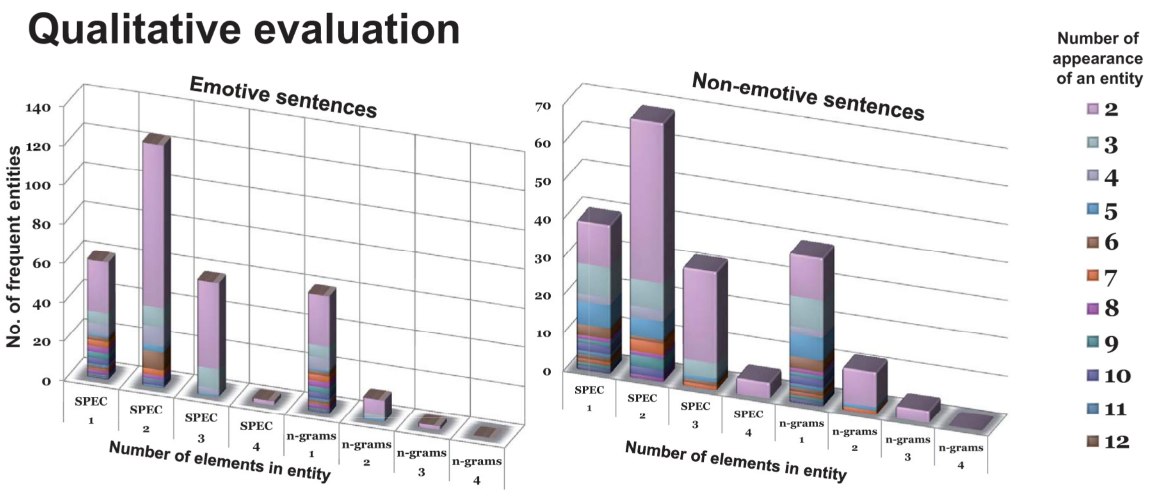


FIGURE 4: Graphs showing differences in numbers of frequent entities (patterns or n-grams) extracted from two datasets.

Except the discovery about the length of combinations sufficient for extraction of frequent patterns, quantitative analysis of patterns extracted from datasets revealed the following. While the number of frequent n-grams was decreasing rapidly with the increase in number of elements, the number of patterns increased for 2-element patterns and then gradually decreased, providing approximately 5 to 20 times more frequent patterns for emotive utterances and 5 to 10 times more patterns for non-emotive utterances (see Table 3). Whether the dataset domain (emotiveness) and language (Japanese) were influential on the overall number of patterns should be an object of further study, however, were able to prove that the pattern based approach does provide more frequent patterns in both of the examined cases. Moreover, qualitative analysis revealed, that n-grams appeared usually with the low occurrence and, excluding unigrams, there were almost no n-grams of higher occurrence then two. On the other hand, for patterns, about one third of the extracted patterns was of occurrence higher then 2 (3 or more). This proves that pattern based language analysis should allow more detailed analysis of language structures than the traditional n-gram approach. Finally, all results were statistically significant on a standard 5% level.

Data sets	n-grams	patterns (without n-grams)	all patterns
	2-element entities		
Emotive sentences	11	113	124
Non-emotive sentences	11	57	68
	3-element entities		
Emotive sentences	2	56	58
Non-emotive sentences	3	28	31
	4-element entities		
Emotive sentences	0	4	4
Non-emotive sentences	0	4	4

TABLE 3: Number of frequent entities extracted (n-grams, non-n-gram patterns and all patterns) from two datasets (collections of emotive and non-emotive sentences). Results presented separately for 2, 3, and 4 element entities.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented SPEC, or *Sentence Pattern Extraction arChitecturte*. The system extracts all frequent sentence patterns from corpora. The extracted patterns are more sophisticated than the ones obtained in a usual n-gram approach, as SPEC does not assume that two subsequent elements of a pattern appear subsequently also in the sentence. SPEC firstly generates all combinations of possible patterns and calculates their pattern statistics (number of occurrences, frequency and weights). SPEC is capable of processing corpora written in any language, as long as they are minimally preprocessed (spacing, POS tagging, dependency structure, etc.). Experiments showed that the method extracts significantly more frequent patterns than the usual n-gram approach. There are two issues needing attention in the near future. Firstly, to make this approach scalable to large corpora (several thousands of sentences of different length), the algorithms need to be further optimized to generate and extract patterns in a faster manner. In pattern generation, a further study in the longest sufficient pattern length will be necessary. Also, applying high performance computing techniques, such as parallel computing should provide much decrease in processing time. Another issue is noise. Although there have been no coherent definition so far of what a noise is in n-gram models, it would be naive to assume that all patterns are always valuable for all applications of the method. One method to deal with this issue should be raising the threshold of frequent patterns (to 3 or higher). Finally, in the near future we plan to apply SPEC to other NLP tasks, such as spam classification, user detection, sentiment analysis [13], cyberbullying detection [16], or lexicon expansion for affect analysis [17] to evaluate SPEC in practice.

Acknowledgments
This research was supported by (JSPS) KAKENHI Grant-in-Aid for JSPS Fellows (Project Number: 22-00358). Authors thank Tyson Roberts and Jacek Maciejewski for their invaluable help with optimizing SPEC algorithms. SPEC repositories are available freely at: http://arakilab.media.eng.hokudai.ac.jp/~ptaszynski/research.htm

REFERENCES

[1] P. F. Brown, P. V. de Souza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. “Class-based n-gram models of natural language”. Computational Linguistics, Vol. 18, No. 4 (December 1992), 467-479, 1992.

[2] Burkhanov. “Pragmatic specifications: Usage indications, labels, examples; dictionaries of style, dictionaries of collocations”, In Piet van Sterkenburg (Ed.). A practical guide to lexicography, John Benjamins Publishing Company, 2003.

[3] S. Chen, J. Goodman, “An empirical study of smoothing techniques for language modeling”, Comp. Speech & Language, Vol. 13, Issue 4, pp. 359-393, 1999.

[4] N. Constant, C. Davis, C. Potts and F. Schwarz, "The pragmatics of expressive content: Evidence from large corpora". Sprache und Datenverarbeitung, 33(1-2):5-21, 2009.

[5] G. Forman. "An extensive empirical study of feature selection metrics for text classification". J. Mach. Learn. Res., 3 pp. 1289-1305, 2003.

[6] P. H. Grice, Studies in the Way of Words. Cambridge (MA): Harvard University Press, 1989.

[7] D. Knight, and S. Adolphs, "Multi-modal corpus pragmatics: The case of active listenership", Pragmatics and Corpus Linguistics, pp. 175-190, Berlin, New York (Mouton de Gruyter), 2008.

[8] D. E. Knuth, The Art of Computer Programming, Volume 4, Fascicle 3: Generating All Combinations and Partitions. Addison Wesley Professional, 2005.

[9] K. Krippendorff, "Combinatorial Explosion", Web Dictionary of Cybernetics and Systems. Princia Cybernetica Web.

[10] T. Kudo. MeCab: Yet Another Part-of-Speech and Morphological Analyzer, 2001. http://mecab.sourceforge.net/ [July 27, 2011].

[11] S. C. Levinson, Pragmatics. Cambridge University Press, 1983.

[12] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. "Text classification using string kernels", The Journal of Machine Learning Research, 2, pp. 419-444, 2002.

[13] B. Pang, L. Lee, S. Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques". In Proc. of EMNLP'02, pp. 79-86, 2002.

[14] P. Pantel and M. Pennacchiotti, "Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations", In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pp. 113-120, 2006.

[15] C. Potts and F. Schwarz. "Exclamatives and heightened emotion: Extracting pragmatic generalizations from large corpora". Ms., UMass Amherst, 2008.

[16] M. Ptaszynski, P. Dybala, T. Matsuba, F. Masui, R. Rzepka, K. Araki and Y. Momouchi, "In the Service of Online Order: Tackling Cyber-Bullying with Machine Learning and Affect Analysis", International Journal of Computational Linguistics Research, Vol. 1 , Issue 3, pp. 135-154, 2010.

[17] M. Ptaszynski, P. Dybala, R. Rzepka K. and Araki, "Affecting Corpora: Experiments with Automatic Affect Annotation System - A Case Study of the 2channel Forum", Proceedings of PACLING-09, pp. 223-228, 2009.

[18] E. Riloff, "Automatically Generating Extraction Patterns from Untagged Text", In Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), pp. 1044-1049, 1996.

[19] B. Roark, M. Saraclar, M. Collins, "Discriminative n-gram language modeling", Computer Speech & Language, Vol. 21, Issue 2, pp. 373-392, 2007.

[20] K. Sasai, "The Structure of Modern Japanese Exclamatory Sentences: On the Structure of the *Nanto*-Type Sentence". Studies in the Japanese Language, Vol, 2, No. 1, pp. 16-31, 2006.

[21] F. Sebastiani. “Machine learning in automated text categorization”. ACM Comput. Surv., 34(1), pp. 1-47, 2002.

[22] C. E. Shannon, “A Mathematical Theory of Communication”, The Bell System Technical Journal, Vol. 27, pp. 379-423 (623-656), 1948.

[23] D. Guthrie, B. Allison, W. Liu, L. Guthrie, Y. Wilks, Y. “A Closer Look at Skip-gram Modelling”. In Proc. Fifth International Conference on Language, Resources and Evaluation (LREC'06), pp. 1222-1225, 2006.

[24] P. P. Talukdar, T. Brants, M. Liberman and F. Pereira, “A Context Pattern Induction Method for Named Entity Extraction”, In Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X), pp. 141-148, 2006.

[26] H. Uchino, S. Shirai, S. Ikehara, M. Shintami, “Automatic Extraction of Template Patterns Using n-gram with Tokens” [in Japanese], IEICE Technical Report on Natural Language Understanding and Models of Communication, 96(157), pp. 63-68, 1996.

INSTRUCTIONS TO CONTRIBUTORS

Computational linguistics is an interdisciplinary field dealing with the statistical and/or rule-based modeling of natural language from a computational perspective. Today, computational language acquisition stands as one of the most fundamental, beguiling, and surprisingly open questions for computer science. With the aims to provide a scientific forum where computer scientists, experts in artificial intelligence, mathematicians, logicians, cognitive scientists, cognitive psychologists, psycholinguists, anthropologists and neuroscientists can present research studies, International Journal of Computational Linguistics (IJCL) publish papers that describe state of the art techniques, scientific research studies and results in computational linguistics in general but on theoretical linguistics, psycholinguistics, natural language processing, grammatical inference, machine learning and cognitive science computational models of linguistic theorizing: standard and enriched context free models, principles and parameters models, optimality theory and researchers working within the minimalist program, and other approaches. IJCL is a peer review journal and a bi-monthly journal.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJCL.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 2, 2011, IJCL appears in more focused issues. Besides normal publications, IJCL intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

IJCL List of Topics:
The realm of International Journal of Computational Linguistics (IJCL) extends, but not limited, to the following:

- Computational Linguistics
- Computational Models
- Computational Theories
- Corpus Linguistics
- Formal Linguistics-Theoretic and Grammar Induction
- Information Retrieval and Extraction
- Language Generation
- Language Learning
- Linguistics Modeling Techniques
- Linguistics Theories
- Machine Translation
- Models of Language Change and its Effect on Lingui
- Models that Address the Acquisition of Word-order
- Models that Combine Linguistics Parsing
- Models that Employ Statistical/probabilistic Gramm
- Models that Employ Techniques from machine learnin
- Natural Language Processing
- Quantitative Linguistics
- Speech Analysis/Synthesis
- Speech Recognition/Understanding
- Spoken Dialog Systems
- Web Information

CALL FOR PAPERS

Volume: 3 - **Issue:** 1 - February 2012

i. Paper Submission: November 30, 2011 **ii. Author Notification:** January 01, 2012

iii. Issue Publication: January / February 2012

CONTACT INFORMATION

Computer Science Journals Sdn BhD
M-3-19, Plaza Damas Sri Hartamas
50480, Kuala Lumpur MALAYSIA

Phone: 006 03 6207 1607
006 03 2782 6991

Fax: 006 03 6207 1697

Email: cscpress@cscjournals.org