TransChat: Cross-Lingual Instant Messaging for Indian Languages

Diptesh Kanojia[1] **Shehzaad Dhuliawala**[1] **Abhijit Mishra**[1]
Naman Gupta[2] **Pushpak Bhattacharyya**[1]
[1]Center for Indian Language Technology, CSE Department
[1]IIT Bombay, India, [2]Yahoo Labs, Japan
[1]{diptesh,shehzaadzd,abhijitmishra,pb}@cse.iitb.ac.in
[2]naman.bbps@gmail.com

Abstract

We present *TransChat*[1], an open source, cross platform, Indian language Instant Messaging (IM) application that facilitates cross lingual textual communication over English and multiple Indian Languages. The application is a client-server IM architecture based chat system with multiple *Statistical Machine Translation (SMT)* engines working towards efficient translation and transmission of messages. *TransChat* allows users to select their preferred language and internally, selects appropriate translation engine based on the input configuration. For translation quality enhancement, necessary pre- and post-processing steps are applied on the input and output chat-texts. We demonstrate the efficacy of *TransChat* through a series of qualitative evaluations that test- (a) The usability of the system (b) The quality of the translation output. In a multilingual country like India, such applications can help overcome language barrier in domains like tourism, agriculture and health.

1 Introduction

In a multilingual country like India which has around 22 official languages spoken across 29 states by more than one billion people, it often becomes quite difficult for a non-speaker of a particular language to communicate with peers following the same language. Be it a non-Indian tourist visiting India for the first time or a farmer from Punjab trying to get tips from a Tamil speaking professor on modern agricultural tools and techniques, communication is often hindered by *language barrier*. This problem has been recognized and well-studied by computational linguists as a result of which a large number of automatic translation systems have been proposed and modified in the last 30 years. Some of the notable Indian Language translation systems include Anglabharati (Sinha et al., 1995), Anusaraka (Padmanathrao, 2009), Sampark (Anthes, 2010) and Sata-Anuvaadak[2] (Kunchukuttan et al., 2014). Popular organizations like Google and Microsoft also provide translation solutions for Indian languages through Google- and Bing-Translation systems. Stymne (2011) demonstrate techniques for replacement of unknown words and data cleaning for Haitian Creole SMS translation, which can be utilized in a chat scenario. But even after so many years of MT research, one can still claim that these systems have not been able to attract a lot of users. This can be attributed to factors like- (a) poor user experience in terms of UI design, (b) systems being highly computational-resource intensive and slow in terms of response time, and (c) bad quality translation output. Moreover, the current MT interfaces do not provide a natural environment to attract more number of users to use the system in an effective way. As Flournoy and Callison-Burch (2000) point out, a successful translation application is one that can *reconcile overly optimistic user expectations with the limited capabilities of current MT technologies*. We believe, a chat application like *TransChat* bridges the current web programming paradigms with MT to provide users a powerful yet natural mode of communication.

[1]http://www.cfilt.iitb.ac.in/transchat/

[2]http://www.cfilt.iitb.ac.in/indic-translator

Our work is motivated by the following factors,

- The ever-increasing usage of hand-held devices and Instant messaging provides us a rich platform to float our application. We can expect a large number of users to download and use it.
- Unavailability of cross lingual instant messaging applications for Indian languages motivates us to build and open-source our application that can be modified by developers as per the needs.

There are several challenges in building such an application. Some of them are,

1. IM users often abbreviate text (especially when the input language is English) to save time. (*e.g. "Hw r u?"* instead of *"How are you?"*). A translation system is often built on top of human-crafted rules (RBMT) or by automatically learning patterns from examples (EBMT) or by memorizing pattens(SMT). In all of these cases, the systems require the input text to be grammatically correct, thereby, making it necessary to deal with ungrammatical input text
2. To provide a rich *User Experience*, the basic requirement of an IM system is that it should transmit messages with ease and speed. Hence, language processing modules should be light-weight in order not to incur delay.

We try to handle these challenges by making use of appropriate language processing modules. The novelty of our work is three fold:

- Our system is scalable, *i.e.*, it allows large number of concurrent user access.
- It handles ungrammatical input through fast efficient text normalization and spell checking.
- We have tried to build a flexible system, *i.e* it can work with multiple translation engines built using different Machine Translation toolkits.

In the following sections we describe the system architecture, UI design and evaluation methodologies.

2 System Architecture

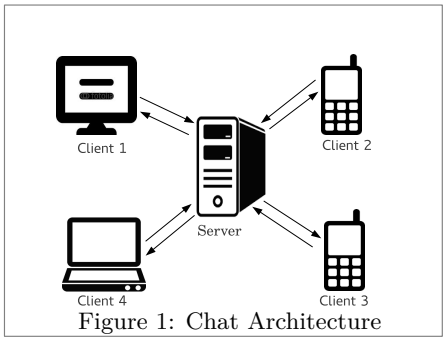

Figure 1: Chat Architecture

Figure 1 shows the system architecture of *TransChat*. Our system consists of a highly scalable chat server ejabberd[3] based on *Extensible Messaging and Presence Protocol(XMPP)*[4] protocol.

It is extensible via modules, which can provide support for additional capabilities such as saving offline messages, connecting with IRC channels, or a user database which makes use of user's vCards. It handles the chat requests, users and message transactions from one user to another.

We build our chat client using publicly available Smack library APIs[5] , which is easily portable to Android devices, thus, providing cross device integration for our chat system.

A user logs in to the server with their respective chat client and gets an option to select a language for his chats. The user then sends a message which is transferred via XMPP protocol to the server where it's processed and then shown to the user at the other end.

Figure 2 shows in detail the processes through which a message passes. The message is processed using the following techniques described in sections 2.1, 2.2, 2.3, and 2.5

2.1 Compression

While chatting, users often express their emotions/mood by stressing over a few characters in the word. For example, usage of words like *thankssss, hellpppp, sryy, byeeee, wowww, goood* corresponds the person being *obliged, needy, apologetic, emotional, amazed,* etc.

As far as we know, it is unlikely for an English word to contain the same character con-

[3]https://www.process-one.net/en/ejabberd/
[4]`http://xmpp.org/xmpp-protocols/`
[5]https://www.igniterealtime.org/projects/smack/

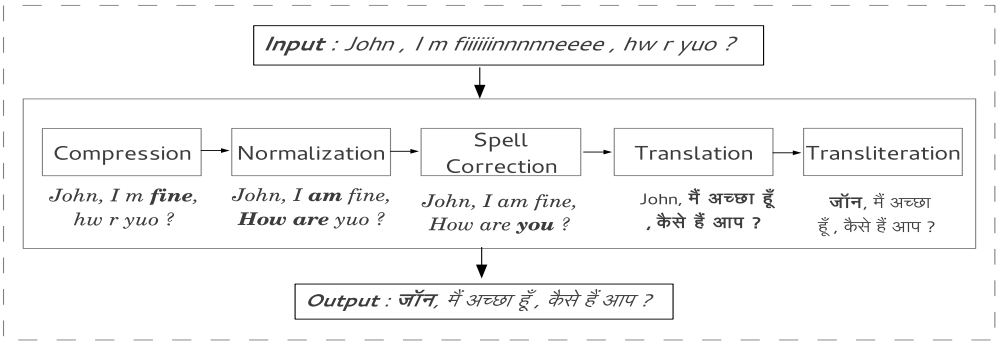

Figure 2: Server Side Processing of *TransChat*

Input Sentence	Output Sentence
I feeellll soooooo gooood	I feel so good
I m veryyyy happyyyyy	I m very happy

Table 1: Sample output of Compression module

secutively for three or more times. We, hence, compress all the repeated windows of character length greater than two, to two characters. For example "pleeeeeeeaaaaaassseeee"is converted to "pleeaassee".

Each window now contains two characters of the same alphabet in cases of repetition. Let n be the number of windows, obtained from the previous step. Since average length of English word (Mayzner, 1965) is approximately 4.9 , we apply brute force search over 2^n possibilities to select a valid dictionary word. If none of the combinations form a valid English word, the compressed form is used for *normalization*.

Table 1 contains sanitized sample output from our compression module for further processing.

2.2 Normalization

Text Message Normalization is the process of translating ad-hoc abbreviations, typographical errors, phonetic substitution and ungrammatical structures used in text messaging (SMS and Chatting) to plain English. Use of such language (often referred as Chatting Language) induces noise which poses additional processing challenges. While dictionary look-up based methods[6] are popular for Normalization, they can not make use of context and domain knowledge. For example, *yr* can have multiple translations like *year, your*.

We tackle this by implementing a normalization system[7](Raghunathan and Krawczyk, 2009) as a Phrase Based Machine Translation System, that learns normalization patterns from a large number of training examples. We use Moses (Koehn et al., 2007), a statistical machine translation system that allows training of translation models.

Training process requires a Language Model of the target language and a parallel corpora containing aligned un-normalized and normalized word pairs. Our language model consists of 15000 English words taken from the web.

Parallel corpora was collected from the following sources :

1. Stanford Normalization Corpora which consists of 9122 pair of un-normalized and normalized words / phrases.

2. The above corpora, however, lacks acronyms and short hand texts like *2mrw, l8r, b4* which are frequently used in chatting. We collected additional data through crowd-sourcing involving peers from CFILT lab[8]. They were asked to enter commonly used chatting sentences/acronyms and their normalized versions. We collected 215 pairs un-normalized to normalized word/phrase mappings.

3. Dictionary of Internet slang words was extracted from `http://www.noslang.com`.

Table 2 contains input and normalized output from our module.

[6] `http://www.lingo2word.com`

[7] Normalization Model: `http://www.cfilt.iitb.ac.in/Normalization.rar`

[8] `http://www.cfilt.iitb.ac.in`

Input Sentence	Output Sentence
shd v go 2 ur house	should we go to your house
u r awesm	you are awesome
hw do u knw	how do you know

Table 2: Sample output of Normalization module

2.3 Spell Checking

Users often make spelling mistakes while chatting. A spell checker makes sure that a valid English word is sent to the Translation system and for such a word, a valid output is produced. We take this problem into account and introduce a spell checker as a *pre-processing* module before the sentence is sent for translation. We have used the JAVA API of Jazzy spell checker[9] for handling spelling mistakes.

An example of correction provided by the Spell Checker module is given below:-

Input: *Today is* ***mnday***
Output: *Today is* ***monday***

Please note that, our current system performs compression, normalization and spell-checking if the input language is English. If the user selects one of the available Indian Languages as the input language, the input text bypasses these stages and is fed to the translation system directly. We, however, plan to include such modules for Indian Languages in future.

If the input language is not English, the user can type the text in the form of Romanized script, which will get transliterated to indic-script with the help of Google's transliteration API. The user can also input the text directly using an in-script keyboard (available in Windows and Android platforms), in which case the transliteration has to be switched off.

2.4 Translation

Translation system is a plug-able module that provides translation between two language pairs. Currently *TransChat* is designed for translation between English to five Indian languages namely, Hindi, Gujarati, Punjabi, Marathi and Malayalam. Translations between these languages are carried out using Statistical Phrase Based Machine Translation paradigm, powered by the Sata-Anuvadak[10] system.

2.5 Transliteration

There is a high probability that a chat will contain many Named Entities and Out of Vocabulary (OOV) words. Such words are not parts of the training corpora of a translation system, and thus do not get translated. For such words, we use Google Transliteration API[11] as a *post processing step*, so that the output comes out in the desired script. Table 3 contains the sample output from our Transliteration module.

3 User Interface

We designed the interface using HTML, CSS, JavaScript and JSP on the front end as a wrapper on our system based on Tomcat web server. We have used Bootstrap[12] and Semantic UI[13] CSS frameworks for the beautification of our interface. The design of our system is simple and sends the chat message using socket to the chat server and receives the response from the server, and shown on the interface.

We use a similar interface for the mobile version of our applications. We have used WebView class in Android, and Windows applications, and UIWebView for iOS. Figure 3 shows a screen-shot of the interface.

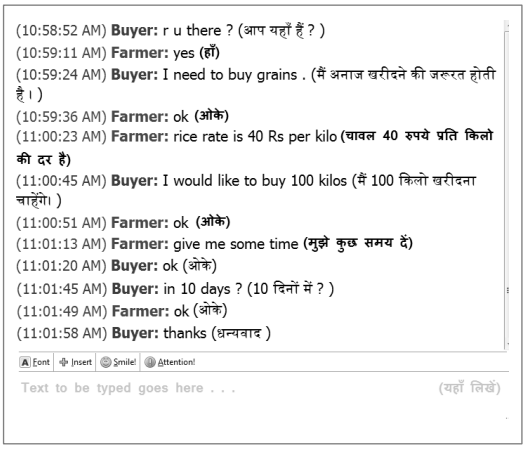

Figure 3: Chat Screenshot

[9] http://sourceforge.net/projects/jazzy/

[10] http://www.cfilt.iitb.ac.in/static/download.html

[11] `https://developers.google.com/transliterate/`

[12] http://getbootstrap.com/

[13] http://semantic-ui.com/

	Input Sentence	Output Sentence
Example 1	Putin अमेरिका आए थे । (Putin amerika aaye the) (Putin visited America)	पुतिन अमेरिका आए थे । (Putin amerika aaye the) (Putin visited America)
Example 2	Telangana भारत का नया राज्य है (telangaana bhaarat ka nayaa rajya hai) (Telangana is a new state of India)	तेलंगाना भारत का नया राज्य है । (telangaana bhaarat ka nayaa rajya hai) (Telangana is a new state of India)

Table 3: Sample output of Transliteration module

	BLEU (Google)	BLEU (SATA)	BLEU Bing	METEOR (Google)	METEOR (SATA)	METEOR (Bing)
Hindi	0.1991	0.1757	0.0497	0.2323	0.1529	0.2217
Marathi	0.088	0.0185	NA	0.1478	0.0843	NA
Punjabi	0.2277	0.0481	NA	0.2084	0.0902	NA
Gujarati	0.0446	0.0511	NA	0.1354	0.1255	NA
Malayalam	0.0612	0.0084	NA	0.1259	0.0666	NA

Table 4: Evaluation Statistics on Normalized input

	P1	P2	P3
P1	1		
P2	.63	1	
P3	.67	.61	1

Table 6: Inter Annotator Agreement for User experience evaluation

4 Evaluation Details

We do a two fold evaluation of our system that ensures the following

- Good user experience in terms of (a) using the system and (b) acceptable translation quality etc.
- The pre- and post-processing steps employed helped enhance the quality of translated chat.

We study the first point qualitatively by employing three human users who are asked to use the system by giving 20 messages each. They have to eventually assign three scores to the system, *Usability score*, *Fluency score of the chat output* and *Adequacy score of the chat output*. Fluency indicates the grammatical correctness where as adequacy indicates how appropriate the machine translation is, when it comes to semantic transfer. We then compute the average Inter Annotator Agreement (IAA) between their scores. Table 6 presents the results. We have obtained a substantial IAA via Fliess' Kappa evaluation.

To justify the second point, we input the un-normalized chat messages to three different translators *viz.* Google, Bing and Sata-Anuvaadak. We then obtain the BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014) translation scores. We then compute the evaluation scores after applying the pre- and post-processing steps explained in section 2. Table 4 and 5 present the scores for processed and un-processed inputs. As we can see, applying pre- and post-processing steps help us achieve better evaluation scores.

We observe a slight inconsistency between BLEU and METEOR scores for some cases. BLEU has been shown to be less effective for systems involving Indian Languages due to *language divergence, free-word order* and *morphological richness* (Ananthakrishnan et al., 2007). On the other hand, our METEOR module has been modified to support stemming, paraphrasing and lemmatization (Dungarwal et al., 2014) for Indian Languages, tackling such nuances to some extent. This may have accounted for the score differences.

5 Conclusion and Future Work

In today's era of short messaging and chatting, there is a great need to make available a multilingual chat system where users can interact despite the language barrier. We present such an Indian language IM application that facilitates cross lingual text based communication. Our success in developing such an application for English to Indian languages is a small step in providing people with easy access to such a chat system. We plan to make efforts towards improving this application to enable chatting in all Indian languages via a

	BLEU (Google)	BLEU (SATA)	BLEU (Bing)	METEOR (Google)	METEOR (SATA)	METEOR (Bing)
Hindi	0.0306	0.0091	0.0413	0.0688	0.053	0.0778
Marathi	0.0046	0.0045	NA	0.0396	0.0487	NA
Punjabi	0.0196	0.0038	NA	0.0464	0.0235	NA
Gujrati	0.0446	0.0109	NA	0.1354	0.06	NA
Malyalam	0.0089	0.0168	NA	0.0358	0.1045	NA

Table 5: Evaluation Statistics on Unnormalized input

common interface. We also plan to inculcate Word Suggestions based on corpora statistics, as they are being typed, so that the user is presented with better lexical choices as a sentence is being formed. We can also perform emotional analysis of the chat messages and introduce linguistic improvements such as euphemism for sensitive dialogues.

Such an application can have a huge impact on communication, especially in the rural and semi-urban areas, along with enabling people to understand each other.

6 Acknowledgment

We would like to thank CFILT, IIT Bombay for granting us the computational resources, and annotation of our experiments. We would also like to thank Avneesh Kumar, who helped us setup the experiment in its initial phase.

References

R Ananthakrishnan, Pushpak Bhattacharyya, M Sasikumar, and Ritesh M Shah. 2007. Some issues in automatic evaluation of english-hindi mt: More blues for bleu. *ICON*.

Gary Anthes. 2010. Automated translation of indian languages. *Communications of the ACM*, 53(1):24–26.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Piyush Dungarwal, Rajen Chatterjee, Abhijit Mishra, Anoop Kunchukuttan, Ritesh Shah, and Pushpak Bhattacharyya. 2014. The iit bombay hindi english translation system at wmt 2014. *ACL 2014*, page 90.

Raymond S Flournoy and Christopher Callison-Burch. 2000. Reconciling user expectations and translation technology to create a useful real-world application. In *Proceedings of the 22nd International Conference on Translating and the Computer*, pages 16–17.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anoop Kunchukuttan, Abhijit Mishra, Rajen Chatterjee, Ritesh Shah, and Pushpak Bhattacharyya. 2014. Sata-anuvadak: Tackling multiway translation of indian languages.

S. Mayzner. 1965. *Tables of Single-letter and Digram Frequency Counts for Various Word-length and Letter-position Combinations*. Psychonomic monograph supplements. Psychonomic Press.

Anantpur Amba Padmanathrao. 2009. Anusaaraka: An approach for mt taking insights from the indian grammatical tradition.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Karthik Raghunathan and Stefan Krawczyk. 2009. Cs224n: Investigating sms text normalization using statistical machine translation. *Department of Computer Science, Stanford University*.

RMK Sinha, K Sivaraman, A Agrawal, R Jain, R Srivastava, and A Jain. 1995. Anglabharti: a multilingual machine aided translation project on translation from english to indian languages. In *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, volume 2, pages 1609–1614. IEEE.

Sara Stymne. 2011. Spell checking techniques for replacement of unknown words and data cleaning for haitian creole sms translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 470–477, Stroudsburg, PA, USA. Association for Computational Linguistics.