SciVerse ScienceDirect

Procedia - Social and Behavioral Sciences 27 (2011) 2 – 11

Procedia
Social and Behavioral Sciences

Pacific Association for Computational Linguistics (PACLING 2011)

Text normalization in social media: progress, problems and applications for a pre-processing system of casual English

Eleanor Clark[a]* and Kenji Araki[a]

[a] *Graduate School of Information Science and Technology, Hokkaido University, Kita 14, Nishi 9, Sapporo 060-0814, Japan*

Abstract

The rapid expansion in user-generated content on the Web of the 2000s, characterized by social media, has led to Web content featuring somewhat less standardized language than the Web of the 1990s. User creativity and individuality of language creates problems on two levels. The first is that social media text is often unsuitable as data for Natural Language Processing tasks such as Machine Translation, Information Retrieval and Opinion Mining, due to the irregularity of the language featured. The second is that non-native speakers of English, older Internet users and non-members of the "in-group" often find such texts difficult to understand. This paper discusses problems involved in automatically normalizing social media English, various applications for its use, and our progress thus far in a rule-based approach to the issue. Particularly, we evaluate the performance of two leading open source spell checkers on data taken from the microblogging service Twitter, and measure the extent to which their accuracy is improved by pre-processing with our system. We also present our database rules and classification system, results of evaluation experiments, and plans for expansion of the project.

© 2011 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of PACLING Organizing Committee.

Keywords: Natural Language Processing; Machine Translation; Social Media; Twitter; Text Normalization

1. Introduction

The rapid expansion of Internet use, electronic communication and user-oriented media such as social networking sites, blogs and microblogging services has led to a rapid increase in the need to understand casual written English, which often does not conform to rules of spelling, grammar and

* Corresponding author. Tel.: +81-11-706-7389; fax: +81-11-709-6277.

E-mail address: eleanor@media.eng.hokudai.ac.jp

1877-0428 © 2011 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of PACLING Organizing Committee.
doi:10.1016/j.sbspro.2011.10.577

punctuation. Despite this, text normalization is commonly seen as cumbersome [1], and remains a somewhat niche topic of research. Studies which attempt to tackle this problem generally use a fully automated, statistical approach [2,3]; however, we propose that a combination of automated and manual techniques is a potentially more useful approach to this problem. Accordingly, our aim is to develop a method which uses automated tokenization, word matching and replacement techniques in combination with a high- quality, large scale, manually compiled database. We present recent progress on this system, CECS (Casual English Conversion System).

CECS has two applications: as pre-processing on noisy input for automated Natural Language Processing tasks such as Machine Translation or Information Retrieval; and as a standalone system for human users, to aid non- native speakers' reading comprehension of informal written English, the irregularity of which may pose a barrier to their positive participation in 21st Century international communications.

This user-oriented educational aspect of CECS is complemented by the inclusion of annotation on linguistic and/or cultural aspects of each word or phrase converted by the system. At present, the system's knowledge base for text replacement is a manually compiled database of 1,043 items, although expansion of the database is constant and regular.

2. Related work

Research aimed at the specific problem of automatically normalizing casual English is relatively rare [4]. While spelling error correction is a well-established area, with initial pattern matching and *n*-gram analysis techniques having improved over the last two decades [5], the range of problems presented by user-generated content in online sources go beyond simple spelling correction; other problems include rapidly changing out- of-dictionary slang, short-forms and acronyms, punctuation errors or omissions, phonetic spelling, misspelling for verbal effect and other intentional misspelling, and recognition of out-of-dictionary named entities [6].

Research on unknown vocabulary items often focuses on the recognition and translation/transliteration of proper names; although Sproat *et al.* [1] included some attempts at automatic expansion of acronyms and abbreviations, slang and casual language were not specifically featured. Sproat et al. note that "text normalization is not a problem that has received a great deal of attention, and it (...) seems to be commonly viewed as a messy chore" [1]. Alexander Clark's work on pre-processing a large collection of the Internet discussion system Usenet's posts, through a straightforward machine learning methodology using generative models and a noisy channel method, made some progress towards handling the type of input discussed here, but faced problems with the quality of the corpus and did not reach the evaluation stage [7]. Aw *et al.* [2] have produced a system for normalizing Short Message Service[b] mobile phone texts, which share many of the characteristics of the casual English focused on in this paper, such as non-standard short-forms of words, creative phonetic or stylistic spelling, and punctuation omission, by creating a parallel corpora of 5,000 raw and normalized English SMS messages and applying a phrase-based SMT model, resulting in significantly more accurate translations when the system's output was passed through commercially available MT systems. The use of a phrase-based model rather than a word-based one incorporates logical contextual information to the translation model and thus improves lexical affinity and word alignment. However, their model is essentially a fairly

[b] Short Message Service, or SMS texts are limited to 160 characters in length, which gives necessity for the creative use of new shortened forms of language.

straightforward SMT system, and was limited by the unavailability of parallel corpora suitable for automated constructing of such a system.

Henriquez *et al.* [3], in their work for the CAW 2.0 project introduced an approach using a *n*-gram based SMT system and were able to produce syntactically correct sentences from input with a high frequency of misspelled words and Internet slang, but again found that their system's effectiveness had "a strong dependency on the dictionary quality and size" and that their "small dictionary is not able to handle all possible abbreviations and terms".

With the rapid expansion of new media, the irregularity of language poses a barrier to automated tasks. Ritter *et al.,* in their modeling of Twitter dialogue acts, found that posts were "often highly ungrammatical, and filled with spelling errors", and resorted to selecting clusters of spelling variations manually [8]. The interest in content of this type, both from researchers and corporations, shows a pressing need for effective text normalization of casual English.

3. Casual English classification system and database

3.1. Casual English classification system

Our Casual English Conversion System (CECS) is designed on the basis that errors and irregular language used in casual English found in social media can be grouped into several distinct categories, and accordingly, a multi-faceted approach will be the most effective way to deal with the problem. The categories used in CECS' database are as follows. **1. Abbreviation (shortform).** Examples: *nite* ("*night*"), *sayin* ("*saying*"); may include letter/number mixes such as *gr8* ("*great*"). **2. Abbreviation (acronym).** Examples: *lol* ("laugh out loud"), *iirc* ("if I remember correctly"), etc. **3. Typing error/ misspelling.** Examples: *wouls* ("*would*"), *rediculous* ("*ridiculous*"). **4. Punctuation omission/error.** Examples: *im* ("*I'm*"), *dont* ("*don't*"). **5. Non-dictionary slang.** This category includes word sense disambiguation (WSD) problems caused by slang uses of standard words, e.g. *that was well mint* ("*that was very good*"). It also includes specific cultural reference or in group-memes. **6. Wordplay.** Includes phonetic spelling and intentional misspelling for verbal effect, e.g. *that was soooooo great* ("*that was so great*"). **7. Censor avoidance.** Using numbers or punctuation to disguise vulgarities, e.g. *sh1t*, *f***,* etc. **8. Emoticons.** While often recognized by a human reader, emoticons are not usually understood in NLP tasks such as Machine Translation and Information Retrieval. Examples: :) (smiling face), <3 (heart).

3.2. Database construction and rules

CECS uses a manually compiled and verified database, currently of a total of 1,043 entries. These entries are either single words or phrases; the trie-type data structure theoretically allows for phrases of unlimited word length, but at present the majority of phrase entries are sets of two or three words. Each entry has been taken from training data which is rich in casual English, including Twitter[c] entries and YouTube[d] comment boards, and meanings have been verified through collaborative user-compiled, user-evaluated resources such as Wiktionary[e] and Urban Dictionary[f]. Database entries comprise of four

[c] http://twitter.com

[d] www.youtube.com

[e] www.wiktionary.org

[f] www.urbandictionary.com

columns: “error word” (the casual English item), “regular word” (the corresponding dictionary English item), “category” (the item’s category as defined in Section 3.1) and “notes” (cultural or linguistic information about the item’s origin, intended for CECS’ human users). Database construction is an ongoing project, and we intend to improve its coverage and quality further. Careful manual editing of the database includes checking to avoid rule conflicts, a common problem in rule-based systems.

3.3. Phrase matching rules

Phrase matching in CECS is an important feature. Firstly, slang phrases constituting more than one word can be matched in the database; secondly, problems regarding word sense disambiguation (WSD) problems can be tackled. When a word exists as a regular English word but is often used in casual English to mean something else, it is not detected by conventional spellcheckers. As an example, the regular English word “rite” is commonly used as a shortened form of “right”. However, it may also be used in its original meaning as “ritual”, as the following example sentences show.

Regular usage: Going to high school is tough, but it is a necessary rite of passage.
Casual usage: seein that ad makes me wanna listen to dat song rite now (*Seeing that advertisement makes me want to listen to that song right now*).

As well as being confusing to non-native readers, this word causes problems to MT applications, which tend to translate it as “ritual”, rendering many casual English sentences difficult to understand after translation. With phrase matching in CECS, common combinations of “rite” which can *only* be used in the sense of “right” can be added into the database. Thus, pre-processing casual English with CECS can improve MT handling of such vocabulary items. Table 1 shows a section of the database entries containing “rite”:

Table 1. Section of database entries containing “rite”

Input	Normalization
is rite	is right
it rite	it right
iz rite	is right
so rite	so right
r rite	are right
rite now	right now
rite away	right away

This approach also proves useful for normalizing numbers which have been used as phonetic substitutions, e.g. “4” for “for”, “2” for “to” or “too”, etc. Whereas it would be obviously inaccurate to automatically convert all instances of the number “4” to “for”, with phrase matching it is possible to convert a high number of occurrences correctly using carefully designed combinations. Thus, we can define the rules for the usage of these items manually, and automatically convert appropriately with CECS. So far, the number of necessary phrase matching rules per vocabulary item differs widely.

While this strategy of addressing WSD cannot yet cover every potential possibility and usage, it is logical that the combinations used are finite and thus can be entered in the database. As more data is

collected, analyzed and more examples gathered, the quality and coverage of the database further increases.

4. System overview

The flow of CECS is shown schematically in Figure 1.

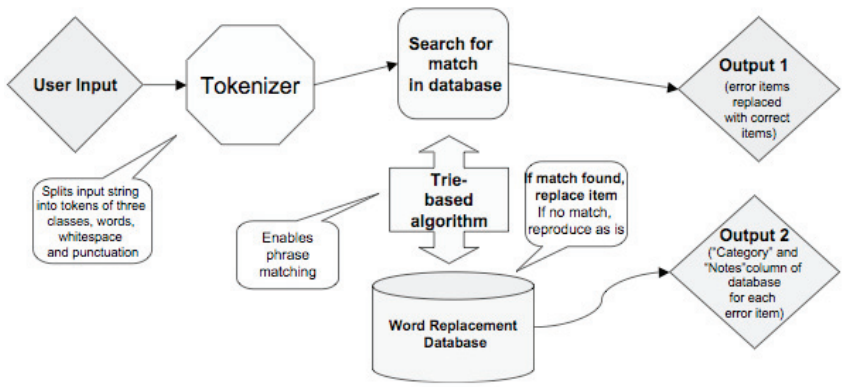

Fig. 1. System flow of CECS

CECS is written in the Python programming language. Firstly, user input is tokenized using a strictly regular grammar defined in PyParsing[g], which defines words and punctuation as separate tokens, and allows combinations. "Main characters" are defined as the letters from a-z and A-Z, numbers 0-9 (in case of spellings which incorporate numbers such as "*gr8*" for "*great*"), and selected punctuation marks which may appear mid-word such as apostrophe ("*don't*"), hyphen ("*mid-word*"), and asterisk for censor avoidance spellings ("*s****"), etc. "Other characters" are defined as all other ASCII characters, and whitespace and carriage returns are defined separately. A token is thus defined here as either a word composed of main characters ("English word") or composed of other characters ("punctuation token").

Tokenized input is then passed through the database to find a match, using a trie-type data structure. The database is recursively loaded into a trie to allow easy item lookup, tokenized by the same tokenizer used for input. Database entries which are a front-anchored substring are allowed, but full matches are not. Using this data structure, multi-word phrase matching is enabled.

When a match is found, the normalized English equivalent is displayed in the user interface in the "Output" pane, and the replaced item's category and notes, where present, are displayed in the "Notes" pane. Tokens not found in the database are passed through unchanged.

5. Experiments

5.1. Overview of previous experiments: CECS on MT input, CECS for human evaluators

[g] http://pyparsing.wikispaces.com/

Previous evaluation experiments were conduction in order to assess CECS' effectiveness as a pre-processing system for Machine Translation (MT) input, and also as a reading aid for non-native readers of English [6].

In testing CECS' as a pre-processor for MT input, 100 sentences from the popular microblogging service Twitter[h] were run through two well-known free MT applications, Google Translate[i] and Systran[j]. The same sentences were then pre-processed with CECS and run through Google Translate and Systran a second time. The quality of the resulting translations was compared by measuring error incidence. The working language pair used was English to Japanese. Of the 100 Twitter sentences, 20 were "known" sentences, i.e., they had been analyzed for error words and those items were pre-entered into the database. The remaining 80 were "unknown" sentences. MT errors were counted manually in two separate categories, "non-translated word" ("NTW") and "wrongly translated word" ("WTW"). An NTW is defined here as the MT application simply reproducing an item as roman letters or numbers, and not converting to Japanese at all. A WTW was defined as a Japanese word that is completely semantic different from the English meaning. In the "unknown" data, with an average sentence length of 15.35 words, there was a decrease in NTW occurrence from 3.34 to 0.86 words per sentence (average of both MT applications). This significant drop showed that CECS' database coverage already gives a reasonable performance.

In evaluating CECS for human users, ten non-native learners of English between the ages of 23 and 64 completed two questionnaires, in which they were asked to assess their understanding of 20 sentences, also taken from Twitter. The first questionnaire used raw input for the sentences, and the second questionnaire used the same sentences after processing by CECS. No participants were allowed to see the corrected sentences until they had submitted the first questionnaire. Rankings were made on a five-point semantic differential scale, as follows:

Question: *How much of the sentence can you understand?*
1. *None at all* ***2.*** *A little* ***3.*** *Some* ***4.*** *Most* ***5.*** *All.*

Evaluators were also asked to give a reason for why they could not understand part or all of each sentence. They were given three choices: vocabulary, grammar and context. Attributing more than one reason to failing to understand a sentence was possible. Evaluators were also asked to assess their level of English comprehension on a scale of 1 (very basic) to 5 (excellent). Overall, average understanding of the 20 sentences increased by exactly one semantic differential point: evaluator comprehension of the sentences averaged at 2.89 for raw input, on the low side of "Some" on the semantic scale, and 3.89 for system output, or slightly lower than "Most" on the semantic scale.

5.2. CECS with spellchecking: GNU Aspell and Hunspell

We have established in Sections 1 and 2 why conventional spellchecking alone is inadequate for tackling the problem of casual English used in social media; namely, that the range of error types is too broad. Conversely, using CECS as a replacement for conventional spellchecking is not useful as the database coverage cannot compete with major spell checkers' dictionaries, and also, "normal" –

---

[h] www.twitter.com

[i] http://translate.google.com

[j] http://www.systranet.com

presumably unintentional - spelling errors (typos and common misspellings) are frequent occurrences in social media data. It was noted that in previous experiments with CECS, a large proportion of unresolved errors fell under the category of typo/spelling errors [6].

Thus, as a solution to this problem, we assumed that the integration of an open-source spellchecker into our system would be the next logical step to improving efficiency. As a preliminary experiment, we compared 80 sentences of Twitter data run through two open source spellcheckers, GNU Aspell[k] and Hunspell[l], firstly unprocessed by CECS and then after pre-processing with CECS. Aspell and Hunspell were chosen due to being the two leading open source spellcheckers at the time of writing. Although Aspell was previously the major open source spellchecker, often featuring in NLP research [4], Hunspell is now the spellchecker of OpenOffice[m] and Mac OS X[n] and it is also used by the Google Chrome[o], Mozilla Firefox[p] and Opera[q] browsers, replacing Aspell in some cases[r].

Data for this experiment was 80 sentences ("tweets") taken from Choudry *et al.*'s [9] Twitter corpus[s]. The sentences had not been used as training data for CECS, and thus there were many out-of-database items which were not corrected by CECS. We hypothesized that using the open source spellcheckers would improve total error correction significantly, and that using CECS as a pre-processing filter would improve the error correction of the spellcheckers. We also hypothesized that Hunspell's performance would be superior, as it has superceded Aspell in many popular applications, as mentioned above.

The experiment process was as follows. Eighty twitter sentences, of a total of 1,208 words in unprocessed form, and then in pre-processed form with a slightly expanded total of 1,253 words, were spellchecked using Aspell and Hunspell respectively. The results were defined into three categories:

a) Regular English Word. A word accepted by the spellchecker's dictionary as a correct English word. Example: "*what* did you say".

b) Resolved Error. A word that was indicated by the spellchecker as an error, and the first replacement candidate offered by the spellchecker was correct. Example: "*wat* did you say*",* replaced by "*what* did you say".

c) Unresolved Error. A word that was indicated by the spellchecker as an error, and the first replacement candidate offered by the spell checker was incorrect. Example: "*wat* did you say", replaced by "*water* did you say".

---

[k] http://aspell.net

[l] http://hunspell.sourceforge.net

[m] http://www.openoffice.org

[n] http://www.apple.com/osx

[o] http://www.google.com/chrome

[p] http://www.mozilla.com/firefox

[q] http://www.opera.com

[r] Opera 10 now using Hunspell : http://my.opera.com/desktopteam/blog/2009/04/03/turbo-in-10 (accessed January 12th 2011)

[s] A 10.5 million "tweet" (Twitter posting) Twitter corpus as compiled and publicly released by Choudhury [9]. The corpus contains tweets from 200,000 unique users collected between 2006 and 2009; the sentences used in our experiment were taken from the September 2009 section of the corpus.

The correctness of replacement candidates was verified by two native speakers. Note that only the first candidate produced by the spellchecker was considered, so even if the second, fifth or tenth candidate was correct, the word would be defined as an Unresolved Error.

5.3. CECS with spellchecking: experiment results

The results of the experiment are shown in Figs. 2 and 3. The first notable outcome is that the use of CECS as a pre-processing system greatly reduced error incidence. The second is that although the two spellcheckers performance is almost identical on the pre-processed data, Aspell seems to have a slight advantage on the unprocessed data.

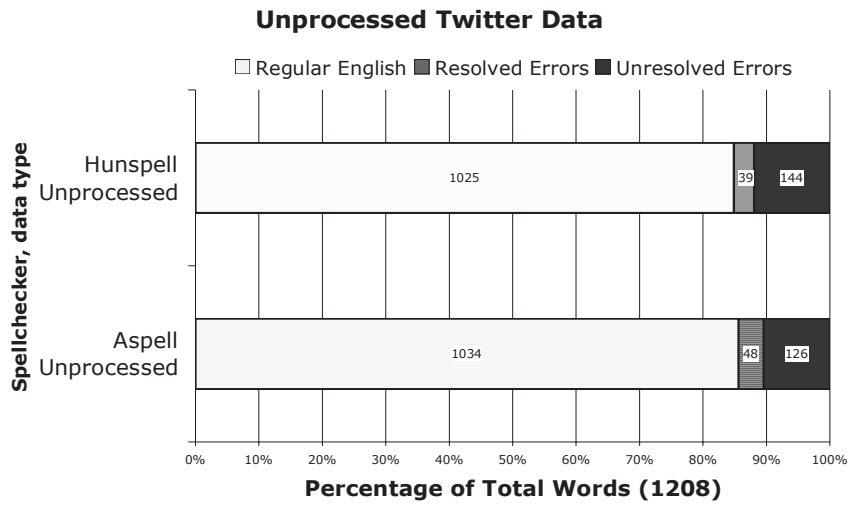

Fig. 2. Unprocessed Twitter data spellcheck results

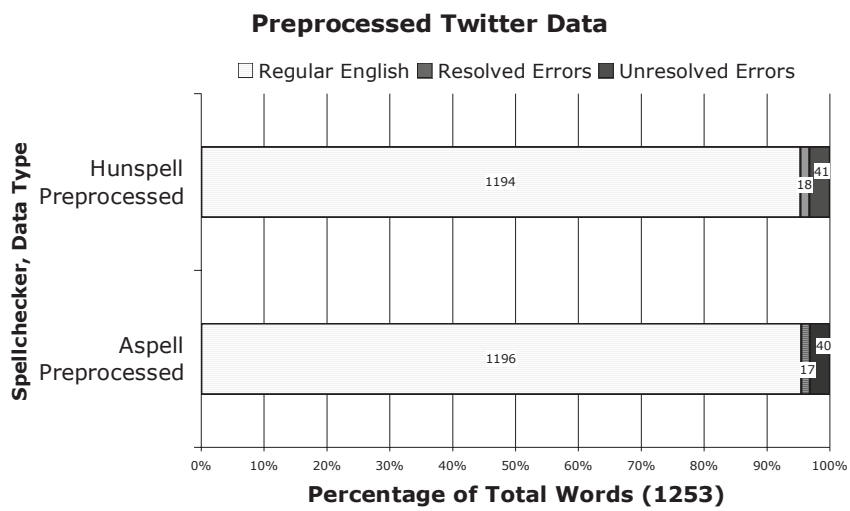

Fig. 3. Pre-processed Twitter data spellcheck results

5.4. CECS with spellchecking: Aspell/Hunspell comparison and evaluation

Broadly speaking, both Hunspell and Aspell usually detected and corrected the same Casual English items, with some significant differences. Although the results seem to indicate that Aspell's performance is superior on the unprocessed data, this is due to error detection as well as error correction, and thus not so clear-cut. For example, Aspell ignores the use of numbers mixed with letters, leading it to treat items such as *hard2tell* ("*hard to tell*") and *sh1t* ("*shit*"), as appeared in our experiment data, as accepted as English words. Hunspell treats number-letter combinations as problems, although did not provide correct candidates for any. This discrepancy in detection partially accounts for higher Unresolved Errors in Hunspell.

Further examples of incomplete error detection are emoticons and single letters. Both spellcheckers ignore punctuation marks, thus emoticons such as ":)" and ":-(" were not flagged; however, emoticons which include letters such as ":p" or ";D" were defined as spelling errors. "Single letters" refers to the use of "*b"* for "*be*", "*r*" for "*are*", "*c*" for "*see*", etc. None of these are flagged as spelling errors, but if used in MT data, for example, they are untranslatable. Normalization with CECS removed these problems in virtually all cases in our experiment data.

Other obvious weaknesses in using the spellcheckers on social media data were WSD problems and non-dictionary slang. For example, neither Hunspell nor Aspell flagged phrases like *rite now*, which causes problems as explained in Section 3.3. WSD also caused problems with apostrophe omission, where the commonly occurring "*cant*" for "*can't*" and "*wont*" for "*won't*" were recognized as correct English words (which presumably would lead them to be translated with the meanings of "hypocrisy" and "habit"). However, apostrophe omission in longer words such as "*havent*" and "*theyre*" tended to be correctly handled. Slang shortforms posed more of a problem. For example, "*im*", frequently used for "*I'm*" (as well as other uses such as an abbreviation for "instant message") was a major stumbling block for Hunspell. Aspell correctly converted "*im*" to "*I'm*", but Hunspell did not even list it as a candidate (candidates were*: mi, um, om, in, i, m, ism, aim, rim, dim, imp, him, vim, Sim, Tim*). The common shortform "*ur*", which can be "*your*" or "*you're*" depending on context, was capitalized and identified as the place name *Ur* by both spellcheckers. However, "*u're*" was correctly identified as "*you're*" by both. Conversely, the frequently occurring "*coz*" for "because" was converted to the proper name "*Cox*" by both Aspell and Hunspell.

The spellcheckers showed their strengths on typing errors that stem from key proximity, for example "*shoukd*" was correctly converted to "*should*" by both Hunspell and Aspell. This kind of typing error is difficult to cover with the CECS database, and here the integration of a spellchecker would be highly beneficial. Hunspell performed well with vowel omission, for example correctly suggesting "*anyhow*" for "*nyhw*", which Aspell did not. Hunspell also performed well with missing final g's, correctly converting "*feelin*", "*hatin*" and "*sayin*" to "*feeling*", "*hating*" and "*saying*", but Aspell consistently suggested "*(verb) in*" as the first candidate. Neither correctly handled the commonly occurring "*goin*" ("*going*"), however, with Hunspell offering "*goon*" and Aspell choosing "*go in*".

Correction of wordplay featuring lengthened vowels was hit-and-miss, with Aspell faring somewhat better than Hunspell. "*Soo*" and "*sooo*" for "*so*" were correctly handled by Aspell but not Hunspell (offering "*sou"* or "*soho*"), but longer examples were not correctly converted by either, e.g. "*oooooohhhh*" prompted Hunspell to offer "*Houyhnhnm, Hohenlohe*" as candidates, whereas Aspell offered the semantically closer "*oohing*".

Overall, with the unprocessed data, the slightly higher Resolved Error count for Aspell was likely due to Aspell's correct conversion of "*im*"; the higher Unresolved Error count in Hunspell was mostly due

to number/letter combinations being flagged as errors, whereas they were ignored by Aspell. However, with the pre-processed data, differences in results were largely insignificant.

This leads us to the question of which spellchecker will be a better choice for integration with our system. Although after pre-processing the difference in results is minimal, considering each spellchecker's strengths and weaknesses with different types of errors, it seems that Hunspell may be slightly more useful, due to its flagging of number/letter mixes and better handling of vowel and final "g" omission.

5. Conclusion

We have presented a discussion of the major problems in processing social media English, the usefulness and possible applications of doing so, and our progress on a system designed for that purpose. We have described an experiment which examines the efficacy of conventional spellcheckers on casual English, and to what extent this could be improved with pre-processing with our system. The results showed that average errors per sentence decreased substantially, from roughly 15% to less than 5%. We concluded from this that the next step for our system should be the integration of an open source spellchecker, which, from its performance in our experiment, will probably be Hunspell. We are confident that the future progress of our project will show further improved results, and that a multi-faceted, integrated approach like ours will prove to be the most effective way of normalizing casual English.

Acknowledgements

The authors gratefully acknowledge support from Hokkaido University Global Centers of Excellence program and from the KDDI Foundation, both of which have made this research possible.

References

[1] Sproat R, Black AW, Chen S, Kumar S, Ostendorf M, Richards C (2001). Normalization of non-standard words. *Computer Speech and Language*, 15(3), 287–333.

[2] Aw A, Zhang M, Xiao J, Su J (2006). A phrase-based statistical model for SMS text normalization. *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, Sydney, Australia, July 2006, 33–40.

[3] Henriquez CA, Hernandez A (2009). A ngram-based statistical machine translation approach for text normalization on chat-speak style communications. *Proceedings of CAW2.0*, Madrid, Spain, August 2009, 1–5.

[4] Wong W, Liu W, Bennamoun M (2007). Enhanced integrated scoring for cleaning dirty texts. *Proceedings of IJCAI 2007 Workshop on Analytics for Noisy Unstructured Text Data*, Hyderabad, India, January 2007, 55–62.

[5] Kukich K (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys,* 24(4), 377–439.

[6] Clark E, Roberts T, Araki K (2010). Towards a Pre- processing System for Casual English Annotated with Linguistic and Cultural Information. *Proceedings of Computational Intelligence 2010*, Hawaii, August 2010.

[7] Clark A (2003). Pre-processing very noisy text. *Proceedings of Workshop on Shallow Processing of Large Corpora,* Lancaster, UK, March 2003, 12–22.

[8] Ritter A, Cherry C, Dolan B (2010). Unsupervised modeling of Twitter Conversations. *Proceedings of HLT-NAACL 2010*, Los Angeles, California, June 2010, 172–180.

[9] Choudhury MD, Lin YR, Sundaram H, Candan KS, Xie L and Kelliher A (2010). How does the sampling strategy impact the discovery of information diffusion in social media? *Proceedings of the 4th International Conference on Weblogs and Social Media*, Washington DC, USA, May 2010.