# DCU-ADAPT: Learning Edit Operations for Microblog Normalisation with the Generalised Perceptron

**Joachim Wagner** and **Jennifer Foster**
ADAPT Centre
School of Computing
Dublin City University
Dublin, Ireland
{jwagner|jfoster}@computing.dcu.ie

## Abstract

We describe the work carried out by the DCU-ADAPT team on the Lexical Normalisation shared task at W-NUT 2015. We train a generalised perceptron to annotate noisy text with edit operations that normalise the text when executed. Features are character $n$-grams, recurrent neural network language model hidden layer activations, character class and eligibility for editing according to the task rules. We combine predictions from 25 models trained on subsets of the training data by selecting the most-likely normalisation according to a character language model. We compare the use of a generalised perceptron to the use of conditional random fields restricted to smaller amounts of training data due to memory constraints. Furthermore, we make a first attempt to verify Chrupała (2014)'s hypothesis that the noisy channel model would not be useful due to the limited amount of training data for the source language model, i.e. the language model on normalised text.

## 1 Introduction

The W-NUT Lexical Normalisation for English Tweets shared task is to normalise spelling and to expand contractions in English microblog messages (Baldwin et al., 2015). This includes one-to-many and many-to-one replacements as in "we're" and "l o v e". Tokens containing characters other than alphanumeric characters and the apostrophe are excluded from the task, as well as proper nouns and acronyms that would be acceptable in well-edited text. (The input, however, does not identify such tokens and unnecessarily modifying them is penalised in the evaluation.)

To make evaluation easier, participants are further required to align output tokens to input tokens, e.g. when the four tokens "l", "o", "v" and "e" are amalgamated to the single token "love", three empty tokens must follow in the output. This is easy for approaches that process the input token by token but may require extra work if the input string is processed differently.

We participate in the constrained mode that allows off-the-shelf tools but no normalisation lexicons and additional data to be used. Furthermore, we do not use any lexicon of canonical English but learn our normalisation model purely from the provided training data.

Our approach follows previous work by Chrupała (2014) in that we train a sequence labeller to annotate edit operations that are intended to normalise the text when applied to the input text. However, while Chrupała uses conditional random fields for sequence labelling, we further experiment with using a generalised Perceptron and with using a simple noisy channel model with character $n$-gram language models trained on the normalised side of the training data to select the final normalisation from a set of candidate normalisation generated from an ensemble of sequence labellers and from selectively ignoring some of the proposed edit operations.

## 2 Experimental Setup

### 2.1 Data Set and Cross-validation

The microblog data set of the shared task contains 2,950 tweets for training and 1,967 tweets for final testing. Each tweet is tokenised and the tokens of the normalised tweets are aligned to the input, allowing for one-to-one, many-to-one and one-to-many alignments.

For five-fold cross-validation, we sort the training data by tweet ID and split it into 5 sets of roughly the same number of tokens. (The number of tweets varies from 579 to 606.) Systems are trained on four sets and tested on the remaining set. Since the sequence labellers require a development set, we split the union of the four sets again into 5 sets to carry out nested cross-validation, training 25 models in total for each system.

## 2.2 Feature Extraction

For extracting recurrent neural network language model features, we use Elman[1] (Chrupała, 2014), a modification of the RNNLM toolkit[2] (Mikolov et al., 2010; Mikolov, 2012) that outputs hidden layer activations. We use the off-the-shelf model from Chrupała (2014)[3]. The input are the characters of the tweet[4] in one-hot encoding. The network has a hidden layer with 400 neurons and it predicts the next byte. Following Chrupała (2014), we reduce the 400 activations to 10 binary features: We select the 10 most active neurons in order and apply a threshold (0.5) to the activation. The value of the $i$-th feature expresses which neuron was $i$-th active and whether its activation was below 0.5, e.g. the first feature states which neuron is most active and whether or not its activation is below 0.5. As there are 400 neurons and 2 possible binarised activations, there are 800 possible values.[5]

Edit operations are extracted from the parallel training data searching for the lowest edit distance and recording the edit operations with dynamic programming. We customise the edit costs function to always postpone insertions to after deleting characters so that each input character can be assigned exactly one edit operation from the set {do nothing, delete character, insert string before character}. To capture insertions at the end of the tweet, we append a NULL byte to all tweets.

The above setup, features and edit operations are identical to Chrupała (2014) to the best of our knowledge. We further add a character class feature {NULL, control, space, apostrophe, punctuation, digit, quote, bracket, lowercase letter, uppercase letter, non-ASCII, other} and a feature indicating whether the character is part of a token that is eligible for editing according to the shared task rules, i.e. whether or not the characters encountered since the last space or start of tweet only are letters, digits, apostrophes and spaces.

## 2.3 Sequence Labelling

For character-level sequence labelling, we try *(a)* Sequor[6] (Chrupała and Klakow, 2010), an implementation of the generalised perceptron (Collins, 2002),[7] with 10 iterations, and *(b)* Wapiti[8] (Lavergne et al., 2010)'s implementation of conditional random fields (Lafferty et al., 2001) using l-bfgs optimisation with a history of 5 steps, elastic net regularisation ($\rho_1 = 0.333$ and $\rho_2 = 0.001$) and no hard limit on the number of iterations. We extend the feature templates of Chrupała (2014)[9] by including our additional two features. The template generates unigram, bigram and trigram character features within a +/- 2 window. All remaining features are included as unigrams of the current value.

Due to the nested cross-validation (see above), Sequor is trained on $64\%$ ($0.8^2$) of the training data, $16\%$ ($0.8 \times 0.2$) is used as development set and 20% (1/5) for testing. For Wapiti, we use only $16\%$ for training (and the remaining $64\%$ for development set) in each cross-validation fold due to memory constraints.[10]

## 2.4 Generating Candidates

We produce candidate normalisations from the edit operations proposed by the sequence model. However, if we allowed each insert and delete operation to be either realised or not, we would produce up to $2^N$ candidates, where $N$ is the number of edit operations. With $N = 140$ (maximum lengths of a tweet), handling these many candidates is not feasible. Instead, we recursively split the sequence of edit operations produced by the sequence labeller into up to eight sections. To find good split points, we propose to minimise

$$\sqrt{|e_L - e_R|} + \max(\{0, 10 - s\})/2 \qquad (1)$$

[1] `https://bitbucket.org/gchrupala/elman`

[2] `http://rnnlm.org/`

[3] `https://bitbucket.org/gchrupala/codeswitch/overview`

[4] More precisely, we process UTF-8 bytes. For the training data, this is the same as characters as the training set does not contain any multi-byte UTF-8 characters.

[5] These RNN-LM hidden layer activation features have been used successfully in text segmentation and word-level language identification (Chrupała, 2013; Barman et al., 2014).

[6] `https://bitbucket.org/gchrupala/sequor`

[7] The generalised perceptron has been shown to match performance of state-of-the-art methods in word segmentation, POS tagging, dependency parsing and phrase-structure parsing (Zhang and Clark, 2011).

[8] `https://wapiti.limsi.fr/`

[9] We thank Grzegorz Chrupała for providing his template and for translating it to the Sequor template format.

[10] With 64%, memory usage grew to over 400 GB over night, causing heavy swap activity on our machines with 256 GB RAM (and 410 GB swap space).

where $e_L$ and $e_R$ are the number of insert or delete operations to the left and right respectively, and $s$ is the number of consecutive no-operations to the left. The first term tries to balance the number of edit operations on each side while the second term introduces a preference to not split clusters of edit operations.

For each section, we either use the edit operations produced by the sequence labeller or do not edit the section. As we split each sequence into no more than eight sections, we produce up to $2^8 = 256$ candidates.[11] Only one candidate, identical to the input, will be produced if there are no delete or insert operations and two candidates will be produced if there is just one delete or insert operation.

In training, we may potentially produce up to $5 \times 256 =$ 1,280 candidates per tweet as the nested cross-validation gives us five sequence labellers per cross-validation run. During testing, up to $25 \times 256 =$ 6,400 candidates may be produced. (The actual maximum number of candidates may be lower when labellers agree on the edit operations.)

### 2.5 Applying Edit Operations

After producing candidate edit operation sequences that use subsets of the edit operations predicted by a sequence model, the edit operations are executed to produce candidates strings for the normalised tweets. As the shared task asks for tokenised output aligned to the input tokens, we apply the edit operations to each token in the following sequence:

1. Apply all edit operations at character positions that correspond to input tokens.

2. Apply insert operations recorded at the space between tokens and at the end of the tweet to the preceding token.

3. Apply delete operations at the space between tokens, moving the contents of the token to the right to the end of the token to the left, leaving behind an empty token. (Delete operations at the end-of-tweet marker are ignored.)

Due to time constraints, we do not attempt to improve the alignment of output tokens to input tokens.

### 2.6 Language Modelling

For language modelling, we train SRILM (Stolcke, 2002) on the normalised tweets of the training data. As we want to build character $n$-gram models and SRILM has no direct support for this, we re-format the candidate strings to make each character a token. To distinguish space characters from token separators, we represent them with double underscores.

### 2.7 Candidate Selection

We use the noisy channel model[12] to select the most plausible source $\hat{s}$ for the observed target $t$ from the set of candidates $S(t)$:

$$\arg\max_{s \in S(t)} P(t|s)P(s) \quad (2)$$

$P(s)$ is provided by the language model (Section 2.6). Standard models give high probability to making few or no edits. However, we trust our sequence models as Chrupała (2014) reported encouraging results. Therefore, we give high probability to using the predicted edit operations. We consider two models for $P(t|s)$:

$$P_1(t|s) = \begin{cases} 0.979 & \text{if all edit operations are used} \\ 0.020 & \text{if} \quad s = t \\ 0.001 & \text{otherwise} \end{cases} \quad (3)$$

and

$$P_2(t|s) = \begin{cases} 1 & \text{if all edit operations are used} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Note that $P_1$ is not a proper probability model as there is never exactly one "otherwise" case but $2^i - 2$ cases where $i$ is the number of sections considered in candidate generation, causing the total to be either 0.999 or between 1.001 and $0.999 + 0.001 \times (2^8 - 2) = 1.253$. $P_2$ effectively excludes the original input and all candidates that use only some but not all of the edit operations suggested by the sequence labellers. Since there are five sequence labellers per cross-validation fold due to nested cross-validation and 25 sequence labellers during testing, $P_2$ effectively selects between 5 or 25 candidates.[13]

[11] Splitting the eight sections again would produce $2^{16} =$ 65,536 candidates.

[12] The noisy channel model has been applied successfully to spelling correction (Kemighan et al., 1990; Wilcox-O'Hearn et al., 2008) and machine translation (Way, 2010), among other areas.

[13] Han et al. (2013) also use a trigram language model for normalisation, but only to reduce a larger candidate set to an

|  | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| WB | 14.70 | 9.97 | 7.91 | 7.31 | **7.19** |
| KN | 14.73 | 9.83 | 7.81 | 7.33 | 7.43 |
| GT | 14.63 | 9.88 | 7.91 | 7.45 | 7.44 |

Table 1: Average language model perplexity over the five cross-validation runs for $n$-gram sizes $n = 2, ..., 6$ and smoothing methods WB = Witten-Bell, KN = Keyser-Ney and GT = Good-Turing. Standard deviation $\sigma \leq 0.23$ for all configurations.

### 2.8 Evaluation Measures

We evaluate our best systems using the evalution script provided by the shared task organisers. It counts:

- The number of correctly modified tokens, i.e. tokens that need to be replaced by a new non-empty token and the system correctly predicts this token.
- Number of tokens needing normalisation, i.e. tokens that are modified in the gold output. However, again, tokens that are to be deleted are ignored, e.g. “l o v e” to “love” counts as one event only despite the replacement of three tokens with empty tokens.
- The number of tokens modified by system, i.e. tokens for which a substitution with a non-empty token is proposed by the system.

Based on these numbers, precision, recall and F1-score are calculated and we select the system and configuration to be used on the test set based on highest average F1-score over the 5 cross-validation runs.

## 3 Results

We use character $n$-gram language models in the noisy channel model for candidate selection. To address sparsity of data that arises when test sentences contain $n$-grams that are rare or unseen in the training data, we try Witten-Bell, Keyser-Ney and Good-Turing smoothing. Table 1 shows average cross-validation perplexity for these three smoothing methods and $n = 2, ..., 6$. Over all five cross-validation folds, the language model that gives the lowest perplexity when trained on the training data and applied to the internal test set is the 6-gram model with Witten-Bell smoothing. This confirms the recommendations in the SRILM documentation to use Witten-Bell smoothing when the vocabulary is small such as when building a character language model.

|  |  | P | R | F1 |
|---|---|---|---|---|
| $P_1$ | W | 83.2% | 37.7% | 51.9% |
| $P_1$ | S | 83.2% | 41.0% | 54.9% |
| $P_2$ | W | **85.9%** | 47.7% | 61.4% |
| $P_2$ | S | 85.7% | **56.1%** | **67.8%** |

Table 2: Average cross-validation results over the five cross-validation runs for transition models $P_1$ and $P_2$, W = Wapiti CRF sequence labeller (trained on only 16% of the training data), S = Sequor generalised perceptron sequence labeller (trained on 64% of the training data), P = precision, R = recall, F1 = F1 measure. Standard deviation $\sigma \leq 0.03$ for all cells.

Table 2 shows cross-validation results for the four systems resulting from the choices between transition models $P_1$ and $P_2$ and using the Wapiti CRF or the Sequor generalised perceptron sequence labeller. The differences are not large in precision but for recall, the model $P_1$ performs poorly. Also the CRF consistently has lower recall than the respective perceptron model. Interestingly, the CRF achieves best precision. On F1-score, the best result is obtained with model $P_2$, which reduces the noisy channel model to selection between sequence modeller hypotheses, together with the Sequor sequence modeller.

On the final test set, our best system using $P_2$ and Sequor has precision 81.90%, recall 55.09% and F1 65.87%, placing it fifth out of six submissions in the “constrained” category.

## 4 Discussion

A possible explanation for the low recall obtained with the $P_1$ model is that this noise model cannot counter the effect that shorter sentences generally receive higher language model probability scores and therefore there is a tendency to reject edit operations that insert additional characters.

Furthermore, we observe that our system often assigns inserted text to the wrong evaluation units, e.g. inserting the string “ laughing out” before the space before “lol” and then replacing second “L” of “lol” with “ud”. This is not wrong on the string

$n$-best list before applying more complex models to token-level candidate selection.

level, but in the token-level evaluation, we make two errors: wrongly appending " laughing out" to the previous token and wrongly normalising "lol" to just "loud" instead of "laughing out loud".

Since the model $P_1$ did not come out best, we cannot reject Chrupała (2014)'s hypothesis that the noisy channel model would not be useful. However, our observations also do not provide much support for this hypothesis as we did not include standard models from previous work (Cook and Stevenson, 2009; Han et al., 2013) in our experiment.

## 5 Conclusions

We trained two sequence modellers to predict edit operations that normalise input text when executed and experimented with applying the noisy channel model to selecting candidate normalisation strings.

Future work should:

- Train the CRF on the full training data, either using a more memory-friendly (but possibly slower) optimisation method or using an even larger machine.
- Experiment with LSTM sequence modelling (Hochreiter and Schmidhuber, 1997; Gers, 2001), which has been applied successfully to speech recognition and caption generation (Graves and Jaitly, 2014; Vinyals et al., 2015).
- Combine models with voting rather than language model score.
- For the noisy channel model, try standard models from previous work (Cook and Stevenson, 2009; Han et al., 2013).
- To better understand the selection preferences of the noisy channel model, compare the F1-score obtained when evaluating against the gold data to the F1-score obtained when evaluating the system output against its own input, i.e. are we biased towards doing nothing?
- Introduce a brevity penalty to counter the effect of selecting short candidate normalisations in the noisy channel model.
- Automatically revise the alignment to input token according to global co-occurrence statistics.
- Carry out a full error analysis of what the system does well and where it fails.

## Acknowledgments

This research is supported by Science Foundation Ireland through the CNGL Programme (Grant 12/CE/I2267) in the ADAPT Centre (www.adaptcentre.ie) at Dublin City University. We thank the anonymous reviewers for their comments on this paper. Furthermore, we thank Grzegorz Chrupała for sharing his feature templates and for his suggestion to try Sequor.

## References

Timothy Baldwin, Marie Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text (WNUT 2015)*, Beijing, China.

Utsab Barman, Joachim Wagner, Grzegorz Chrupała, and Jennifer Foster. 2014. DCU-UVT: Word-level language classification with code-mixed data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching. EMNLP 2014, Conference on Empirical Methods in Natural Language Processing*, pages 127–132, Doha, Qatar, October. Association for Computational Linguistics.

Grzegorz Chrupała and Dietrich Klakow. 2010. A named entity labeler for German: Exploiting Wikipedia and distributional clusters. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Grzegorz Chrupała. 2013. Text segmentation with character-level text embeddings. In *Proceedings of the ICML 2013 Workshop on Deep Learning for Audio, Speech and Language Processing*, Atlanta, GA, USA. `https://sites.google.com/site/deeplearningicml2013/accepted_papers`.

Grzegorz Chrupała. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 680–686, Baltimore, Maryland, June. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings* *of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP'02)*, pages 1–8, Morristown, NJ, USA, July. Association for Computational Linguistics.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 71–78, Boulder, Colorado, June. Association for Computational Linguistics.

Felix Gers. 2001. *Long Short-Term Memory in Recurrent Neural Networks*. Ph.D. thesis, École Polytechnique Fédérale de Lausanne, Département d'Informatique, Lausanne. Switzerland. `http://www.felixgers.de/papers/phd.pdf`.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In Eric P. Xing and Tony Jebara, editors, *Proceedings of The 31st International Conference on Machine Learning*, volume 32 of *JMLR Workshop and Conference Proceedings*. `http://jmlr.org/proceedings/papers/v32/`.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST) - Special section on twitter and microblogging services, social recommender systems, and CAMRa2010: Movie recommendation in context archive*, 4(1):5:1–5:27. doi 10.1145/2414425.2414430.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780. doi:10.1162/neco.1997.9.8.1735.

Mark D. Kemighan, Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In Hans Karlgren, editor, *COLING-90: Papers presented to the 13th International Conference on Computational Linguistics on the occasion of the 25th Anniversary of COLING and the 350th Anniversary of Helsinki University, Volume 2*. `http://www.aclweb.org/anthology/C/C90/`.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale crfs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513, Uppsala, Sweden, July. Association for Computational Linguistics.

T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, Makuhari, Chiba, Japan. International Speech Communication Association (ICSA). `http://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf`.

Tomáš Mikolov. 2012. *Statistical Language Models based on Neural Networks*. Ph.D. thesis, Brno University of Technology, Faculty of Information Technology, Department of Computer Graphics and Multimedia, Brno, Czech Republic. `http://www.fit.vutbr.cz/~imikolov/rnnlm/thesis.pdf`.

Andreas Stolcke. 2002. SRILM — an extensible language modeling toolkit. In John H. L. Hansen and Bryan Pellom, editors, *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP2002)*, volume 2, pages 901–904, Baixas, France. International Speech Communication Association (ISCA).

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. `http://arxiv.org/abs/1411.4555`, to appear in Computer Vision and Pattern Recognition.

Andy Way. 2010. Machine translation. In Alexander Clark, Chris Fox, and Shalom Lappin, editors, *The Handbook of Computational Linguistics and Natural Language Processing*, pages 531–573. Wiley Blackwell, Chichester, UK, July.

Amber Wilcox-O'Hearn, Graeme Hirst, and Alexander Budanitsky. 2008. Real-word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing - 9th International Conference, CICLing 2008, Haifa, Isarael, February 17–23, 2008 - Proceedings*, volume 4919/2008, pages 605–616. Springer Berlin/Heidelberg, Germany. 2006 draft version available on http://ftp.cs.toronto.edu/pub/gh/WilcoxOHearn-etal-2006.pdf.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.