An Approach to Unsupervised Text-Normalization

Tobias Elßner
Hauptseminar Unsupervised Learning

Abstract

This paper replicates and partially extends the unsupervised approach to text normalization described in [Sridhar, 2015]. Although it is not able to reproduce their results, it points out obstacles for successful text normalization, and in the end presents a novel attempt to find correspondences between embeddings of different vocabularies and sizes.

1 Introduction

With the increasing importance of social media data, text normalization gains more and more relevance. The objective of text normalization exceeds the one of spelling correction. Spelling correction works mainly on a single-word level, although there are cases of spelling errors, where a resolution needs contextual information. For instance, *pice*, not being a regular english word, could be mispelled, among more possibilities, *rice*, *price*, *piece*, *pace*, *pick*, and can only be resolved by context. What sets text normalization apart from spelling correction is the extension tonormalize rigorous abbreviations and multiword expressions, mostly internet *slang*. Examples here would be *rofl* (abbreviation for *rolling on floor laughing*), *ppl* (for *people*), or including similar sounding numbers, like in *u2* and *4u* (short forms for *you too* and *for you*, respectively).
While classical rule-based techniques and modern sequence-to-sequence alignement algorithms suffer from the necessity of trained experts and hand-annotated training data, unsupervised methods overcome this downside by finding similarities in unnormalized and canonical data automatically.

2 Related Work

The first intuitive approach would be to compile a finite-state-lexicon from unnormalized to normalized forms. Such systems usually have the disadvantage of steadily requiring updates and trained experts to compile them. Also, in many cases there is more than one possible solution, which makes a purely rulebased program hardly usable. [Beaufort et al., 2010] overcome this with a hybrid approach by using a statistical finite state transducer. While the statistical component makes the approach practical, its contextual scope remains limited, because the number of necessary states would eventually explode, especially, if the FST has character transitions.
With an increasing amount of annotated corpora being available, machine translation techniques are becoming usable (cf. [Aw et al., 2006], [Pennell and Liu, 2011]). Here, the string with the highest probability is decoded from an unnormalized sentence through a noisy channel. [Pennell and Liu, 2011] are operating on character, [Aw et al., 2006] on phrase level. In their most successful setups, [Pennell and Liu, 2011] score a precision of 63.3 % (for both single words and abbreviations), and [Aw et al., 2006] reach up to 0.8236 with the `BLEU` trigram metric.
Other, unsupervised approaches, are on the rise. [Gouws et al., 2011] create co-occurrence vectors for unnormalized and normalized strings, weight their entries with pointwise mutual information to filter uninformative contexts, such as determiners, and compare them by cosine similarity. [Yang and Eisenstein, 2013] uses a statistical log-

likelihood model, in which parameters, normally given from parallel corpora, are estimated.
[Hassan and Menezes, 2013] implement a bipartite graph of normalized and unnormalized words in one set and their common contexts in another set of vertices. The edges are weighted according to their number of co-occurrences. Starting at a node of an unnormalized form, they identify its corresponding normalized counterpart via a random walk.
[Gouws et al., 2011] report a word error rate of 5.6%, [Yang and Eisenstein, 2013] present a precision and recall of 82.09%, and [Hassan and Menezes, 2013] give a precision of 92.43% and a recall of 56.4%. Compared to the other unsupervised systems, the word error rate of [Gouws et al., 2011] is rather low, because they evaluate only on unnormalized single words, for instance *r* for *are*.

3 Method

One key problem that applications face is illustrated by the introductory example, *pice*. The exemplary words listed there have a Levenshtein-distance of one. Generating *all* possibilities up to a distance n would lead to an intractable task. A list of words with distance one would already have a size of 286: $26 \cdot 6$ character insertions at the front, in the middle, and at the end, $25 \cdot 5$ character exchanges, and 5 possible deletions.
Solely relying on string similarity leads to an explosion of possible candidates. Therefore, this number needs to be limited to *meaningful* alternatives.
Following [Sridhar, 2015], the mapping from unnormalized to normalized forms is based on semantic, lexical, and syntactic information. In order to select semantically similar normalized forms, word embeddings from unnormalized and 'canonical' texts are compared by cosine similarity:

$$cos(e_u, e_c) = \frac{\sum_{i=1}^{d} e_n[i] \cdot e_c[i]}{\sqrt{\sum_{i=1}^{d} e_n[i] \cdot \sum_{i=1}^{d} e_c[i]}} \tag{1}$$

Here, d is the embedding dimension, e_u the embedding unnormalized, and e_c the embedding of canonical texts. The cosine similarity computes the cosine of the angle between two vectors. Bounded in the intervall [-1, 1], it returns -1, if the two vectors point exactly to the opposite direction, 0, if the vectors are orthogonal to each other, and 1, if the vectors point exactly into the same direction.
To account for misspellings, the following lexical distance is employed:

$$lex\_sim(s_u, s_c) = \frac{lcsr(s_u, s_c)}{ld(s_u, s_c)}, \tag{2}$$

where

$$lcsr(s_u, s_c) = \frac{lcs(s_u, s_c)}{max\_len(s_u, s_c)} \tag{3}$$

In particular, $lcsr(s_u, s_c)$ denotes ratio of the longest common substring divided by the length of the longer one of both strings. $ld(s_u, s_c)$ refers to the levenshtein distance computed between the consonant skeletons of the two strings. Doing so results in a more coarse-grained measurement, making it more robust towards typing errors in the vowel structure of English words. To avoid division by zero, it is set to `1` by default.
Having defined the distance measures, the procedure to obtain a lexicon is as follows:

1. **Selecting *semantically* similar words**
   For each normalized word embedding, select the top-K unnormalized neighbours according to their cosine similarity. The results are saved in a dictionary.

2. **Weighting by edit distance** Next, between all normalized entries and their neighbours, the edit distance is calculated, and added to each ⟨`key, value`⟩ pair.

3. **Inversion of the dictionary** Now, the map can be inverted, such that the unnormalized forms point to their possible normalized counterparts.

The reason, why it is not recommended to compute the top-K *normalized* neighbours for the unnormalized forms, is that there are much more possible unnormalized words corresponding to *one* normalized word than the other way round.

The final step involves an n-gram language model, with Kneser-Ney-smoothing, to ensure a syntactically sound normalized string. Differing from the original approach, this project uses *modified* Kneser-Ney-smoothing ([Chen and Goodman, 1999], cf. Equation 4), testing 3-, 4-, and 5-grams:

$$p_{kn}(w_i \mid w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i}))}{\sum_{w_i} c(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1}) \cdot p_{kn}(w_i \mid w_{i-n+2}^{i-1}) , \tag{4}$$

with

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D_1 N_1(w_{i-n+1}^{i-1}\cdot)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{D_2 N_2(w_{i-n+1}^{i-1}\cdot)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{D_{3+} N_{3+}(w_{i-n+1}^{i-1}\cdot)}{\sum_{w_i} c(w_{i-n+1}^{i})}. \tag{5}$$

$N_1(w_{i-n+1}^{i-1})$, $N_2(w_{i-n+1}^{i-1})$, and $N_{3+}(w_{i-n+1}^{i-1})$ are defined as

$$N_1(w_{i-n+1}^{i-1}) = \mid \{w_i : c(w_{i-n+1}^{i} = 1)\} \mid, \tag{6}$$

$$N_2(w_{i-n+1}^{i-1}) = \mid \{w_i : c(w_{i-n+1}^{i} = 2)\} \mid, \tag{7}$$

and

$$N_{3+}(w_{i-n+1}^{i-1}) = \mid \{w_i : c(w_{i-n+1}^{i} \geq 3)\} \mid . \tag{8}$$

$D(c)$ is

$$D(c) = \begin{cases} 0 \text{ if } c = 0 \\ D_1 \text{ if } c = 1 \\ D_2 \text{ if } c = 2 \\ D_3 \text{ if } c \geq 3. \end{cases} \tag{9}$$

The values for D_1, D_2, and D_{3+}, are calculated as follows:

$$D_1 = 1 - 2Y\frac{n_2}{n_1}, \tag{10}$$

$$D_2 = 2 - 3Y\frac{n_3}{n_2}, \tag{11}$$

$$D_{3+} = 3 - 4Y\frac{n_4}{n_3}, \tag{12}$$

with

$$Y = \frac{n_1}{n_1 + 2n_2}. \tag{13}$$

n_1, n_2, n_3, and n_4 are the number of n-grams occurring exactly once, twice, three or four times.

Using n-grams up to order five resembles a workaround to the original approach, which uses an annotated `SMS`-corpus and therefore embeddings from common multi-word phrases, such as *lol* and *laughing out loud*, respectively. Since this project has no such corpus at hand, unnormalized words in the dictionary, whose normalized suggestions have an edit distance below a certain cutoff, are seen as abbreviations, where each character stands for the first letter of a word. The multiword phrase is then recovered by taking the sequence of words starting with the letters that have the highest n-gram probability. The cutoff point is set to the boundary to the lowest 5% of all edit distances sorted in ascending order.

Five is chosen as the highest order for the n-gram model, because *rofl* (standing for four words) is to the knowledge of the author the longest common abbreviation, while still being a manageble length for n-gram statistics.

To ensure an efficient normalization process, the calculation of the most likely n-gram sequence is done in a greedy manner: For each position in the sequence, the most likely word is chosen, based on the current history.

Although the idea of having a dictionary appears similar to the finite-state approach described in the previous section, it is worth noting that in this case the insertion of new entries is much easier, and does not depend on skilled experts who are familiar with social media slang. New terms, which emerge constantly on the web, just need to be vectorized, and then included following the steps above.

4 Data

The data used here is threefold.

Embedding data from `Twitter` and normalized texts comes from[Pennington et al., 2014].

[Sridhar, 2015] had access to a corpus of human-annotated SMS chats, which was unfortunately

not available for this project. In order to replicate their results approximately, the spoken data of the `Open American National Corpus (OANC)` [Ide and Suderman, 2004] is employed. To compensate for missing multiword-embeddings used by [Sridhar, 2015], multiple n-gram experiments with Kneser-Ney smoothing [Chen and Goodman, 1999] are conducted. The size of n ranges from three to five, whereas in the original publication, a simple trigram model with Kneser-Ney- discounting was implemented. By doing so, the semantic knowledge given by the phrase-vectors is lost, while attempting to normalize text by extending surface (i.e., *n-gram*) information. Spoken data was selected because its text structure is thought to be similar to the test data, namely, `Twitter` posts from the SharedTask[1]. Spoken data consists of the `Charlotte Narrative and Conversation Collection (CNCC)`, containing "95 narratives, conversations and interviews representative of the residents of [...] North Carolina communities" [2], and the `LDC Switchboard corpus`, comprising "of 2320 spontaneous conversations averaging [six] minutes in length and comprising about [three] million words of text, spoken by over 500 speakers of both sexes from every major dialect of American English"[3]. The amount of text (3,217,772 words in total) seems sufficient for a representative language model.
Lastly, the test data comes from the `SharedTask` and comprises of 1967 test sentences.

5 Experiments

In order to be able to run the implementation on a customary PC, the data was reduced to the most frequent 15,000 normalized and 45,000 unnormalized words. Words starting with @ and # are left out, as those stand for proper user names and special topics which do not need to be normalized. Following the procedure described in [Sridhar, 2015], the top *30* neighbours according to the cosine similarity get selected. This is done for different embedding dimensions: 50 and 100. Each dimension size is then paired with one n-gram order, with n being three, four, and five.

6 Results

After the first test runs, it turned out that using the cutoff-point to the lowest 5% of all calculated lexical similarities to detect multi-word expressions does not perform well. The reason is because due to the restriction to the top 45,000 unnormalized `Twitter` terms, abbreviations such as *lol* are not contained in the dictionary.
As a remedy, each word (not starting with @ or #) with less than or equal to four characters is tested for being an abbreviation for a multiword-phrase with method described above. The probability of the sequence is then compared with the probability of the most likely looked-up candidate and the plain (non-normalized) version of the term.
Normalizations annotated by the program (meaning differences between the output of the system compared to its input) are then used as basis for the statistics:

$$\text{Precision} = \frac{\#\text{ Correct Normalizations}}{\#\text{ All identified Normalizations}} \tag{14}$$

$$\text{Recall} = \frac{\#\text{ Correct Normalizations}}{\#\text{ All relevant Normalizations}} \tag{15}$$

$$\text{F-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{16}$$

It is worth noting that capitalization is ignored. Multi-word phrases are treated as one token.
In the following, the results are presented.

	3-gram	4-gram	5-gram
Precision	0.0	0.0	0.0
Recall	0.0	0.0	0.0
F-Score	-	-	-

Table 1: Results for Embedding Dimension = 50

[1]`https://noisy-text.github.io/norm-shared-task.html#task`
[2]`http://www.anc.org/data/oanc/contents/#charlotte`
[3]`http://www.anc.org/data/oanc/contents/#switchboard`

	3-gram	4-gram	5-gram
Precision	0.0	0.0	0.0
Recall	0.0	0.0	0.0
F-Score	-	-	-

Table 2: Results for Embedding Dimension = 100

Both tables show the same results for all conducted experiments. Neither the embedding size, nor the n-gram order affect the performance of the method. In fact, the proposed method does not affect the unnormalized input.
The next section takes a closer look at the problems of the approach.

7 Discusssion

When taking a closer look on the normalizations carried out by the implementation, it becomes apparent that most of the test data remaines unchanged. Especially two error sources shall be examined.

1. **Embedding Data**
   First of all, the not explicitly spelled out assumption was that the enumeration of the dimensions in both the normalized and the data from `Twitter` appears to be false. The top cosine similarities for dictionary entries do not seem semantically related. For example, *weno*, probably standing for *we know*, has among its closest neighbours (50-dimensional embeddings) *yemen*, *abuse*, *juvenile*, *oregon*, and *dogs*. Another unnormalized entry, *maken*, probably slang for *making*, has among its nearest neighbours (100-dimensional embeddings) *lake*, *magic*, *medalist*, and *practiced*.
   One hint for this problem is also the fact that, although the top 25 nearest cosine-neighbours should be selected for canonical words, sometimes less than 25 words are found. In those cases, the cosine distance to most unnormalized words is probably close to the lower bound, -1, so they do not get selected. This is possible if the `Twitter` and normalized embeddings do not bear the same meanings in their dimensions.
   The last section gives an unsupervised attempt to create correspondences between embeddings of different origins.

2. **Data Restriction**
   Another problem poses the restriction to the top 45,000 terms from `Twitter`. Compared to the size of the `Twitter` data set comprising of 27 billion words, this is an infinitesimal small amount. Given that the provided test data is very diverse, it becomes clear why the results are that bad.

3. **Background Corpus**
   Also, the choice of the spoken language section of `OANC` turned out to be problematic. Although it fulfills the checklist - spontaneous speech, diverse speakers, and partially contemporary -, it does not contain, for instance, the phrase *laughing out loud*. Efforts to compensate this deficiency with a performative modified Kneser-Ney smoothing, have not been successful.

4. **Greedy Calculation of n-gram Sequences**
   The last point of concern is the decision to calculate the most likely sequence of words in an abbreviated phrase (covered by characters) in a greedy fashion. Using dynamic programming, local optima could be evaded, which in some cases might have lead to wrong normalizations.

That being said, it looks like as if the difficulties of this project have their origins rather in the selection of the data, than in the actual program.
The field of text normalization is becoming a major area in the analysis of social media, which makes it an important task to further explore it.

8 Outlook

After discovering the missing correspondence between normalized and `Twitter` embeddings, this project seeked to find a way to determine how the dimensions from one embedding are related to the dimensions of the other. As a goal, a transition matrix $T \in \mathbb{R}^{e_t \times e_n}$, where e_t denotes the embedding size of the `Twitter` terms and e_n the embedding size for normalized texts, should be computed. An entry T_{ij}

stands consequently for the weight with which the ith dimension from `Twitter` embeddings correspond to the jth dimension from normalized embeddings. The idea is that embedding nodes with a similar incoming weightings are similar to each other. Since the dimensions do not need to be enumerated in the same way (it depends on the assignment of words to one-hot vectors on the input side, which is arbitrary), and the optimal embedding sizes do not have to be equal, classical distance measures between vectors cannot be applied.

That is why T is calculated iteratively in PageRank-style [Brin and Page, 1998], with the difference being that here PageRank is run on edges, not on vertices. The degree of association between two embedding nodes depends on the similarity of all combinations of incoming edges, multiplied by the degree of association of the original vertices in the input layer. To force the entries in T to converge, the mutliplication is normalized by the sum of all similarities between the weights outgoing from the vertices of the original vertices in the input layer.

Let $Sim(x, y)$ be a similarity measure between two real numbers x and y. The closer x and y are together, the higher is the outcome of $Sim(x, y)$. Let further $\bar{T}$ and $\hat{T}$ denote the transition matrices between nodes from the input ($\bar{T}$) and embedding layer ($\hat{T}$). Superscript t stands for the tth iteration of the algorithm. $M^{\mathtt{T}} \in \mathbb{R}^{i_t \times e_t}$ denotes the weight matrix of the neural network between the input and embedding layer in the case of the `Twitter` embeddings, and likewise $M^N \in \mathbb{R}^{i_n \times e_n}$ for the weight matrix between the input and embedding layer of the normalized embeddings.

Then

$$\hat{T}_{ij}^{(t+1)} = \sum_{k=1}^{i_t} \sum_{l=1}^{i_n} \frac{Sim(M^{\mathtt{T}}_{ki}, M^N_{lj})}{\sum\limits_{i'=1}^{e_t} \sum\limits_{j'=1}^{e_n} Sim(M^{\mathtt{T}}_{ki'}, M^N_{lj'})} \cdot \bar{T}_{kl}^{(t)} \quad (17)$$

computes the association between `Twitter` embedding dimension i and normalized embedding dimension j at iteration $t + 1$, and

$$\bar{T}_{ij}^{(t+1)} = \sum_{k=1}^{e_t} \sum_{l=1}^{e_n} \frac{Sim(M^{\mathtt{T}}_{ik}, M^N_{lj})}{\sum\limits_{i'=1}^{i_t} \sum\limits_{j'=1}^{i_n} Sim(M^{\mathtt{T}}_{i'k}, M^N_{j'l})} \cdot \hat{T}_{kl}^{(t)} \quad (18)$$

analogously calculates the association between `Twitter` input dimension i and normalized input dimension j, at the $t+1$th iteration. This is done, until the values in both matrices reach convergence, given some tolerance, e.g. 0.01.

A similarity function could for instance be

$$Sim(x, y) = \frac{1}{1 + \| x - y \|_2}, \quad (19)$$

which is bounded between $]0, 1]$, with one being the outcome if $x = y$, and close to zero, if x and y are very distant.

Similar to PageRank, where the sum of the scores for all vertices is one, the sum of all entries for both $\hat{T}$ and $\bar{T}$ is one.

The transition matrices can be optimized by taking their values as initialization for a neural network. It could be trained by translating unnormalized forms to normalized terms and back to unnormalized forms, and comparing those to the original unnormalized form fed originally into the network. With the error obtained by this comparison, the matrix weights could be adjusted.

References

[Aw et al., 2006] Aw, A., Zhang, M., Xiao, J., and Su, J. (2006). A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 33–40. Association for Computational Linguistics.

[Beaufort et al., 2010] Beaufort, R., Roekhaut, S., Cougnon, L.-A., and Fairon, C. (2010). A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 770–779. Association for Computational Linguistics.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117.

[Chen and Goodman, 1999] Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

[Gouws et al., 2011] Gouws, S., Hovy, D., and Metzler, D. (2011). Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 82–90. Association for Computational Linguistics.

[Hassan and Menezes, 2013] Hassan, H. and Menezes, A. (2013). Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1577–1586.

[Ide and Suderman, 2004] Ide, N. and Suderman, K. (2004). The american national corpus first release. In *LREC*.

[Pennell and Liu, 2011] Pennell, D. and Liu, Y. (2011). A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 974–982.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Sridhar, 2015] Sridhar, V. K. R. (2015). Unsupervised text normalization using distributed representations of words and phrases. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 8–16.

[Yang and Eisenstein, 2013] Yang, Y. and Eisenstein, J. (2013). A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 61–72.