Word reordering algorithm for poetry analysis

V B Barakhnin and I S Pastushkov
Institute of Computational Technologies of SB RAS, Lavrentiev av. 6, Novosibirsk, 630090, Russian Federation
E-mail: bar@ict.nsc.ru, pas2shkov.ilya@gmail.com

Abstract. Automatic poetry analysis is usually examined as subtask of natural language processing but approaches which are good for prose texts are ineffective for poetry due to free semantic characteristics. It allows us to use large corpus for training words order and rather small text corpus amount. In this paper we suggest the way to adapt poetry texts to prose ones by changing word order without losing of model and also use word2vec features to improve the classifier work.

1. Introduction

Many of European poetic traditions are characterized by the usage of word order which is different from the usual prose speech. The origins of this phenomenon occurred in epic tales, for example, the early-English epic poem “Beowulf” [1] or old Finnish “Kalevala” [2]. This tradition is continued in more contemporary poetry like Lorelei Heine [3] or in Chaucer’s works [4]. Certainly, the old English language of Beowulf is significantly different from the English one that evolved after the Norman conquest of England. In later English poetry, represented by W. Shakespeare or W. Blake, there are minor changes in the word order, so in this respect their poems are similar with prose texts. Most of the text analysis tools were created specifically for the English language, as for the most common language among researches in this field and as the language of international scientific communication. English is analytical language with a rather strict word order. That's why the computational linguistics algorithms, based on the formal languages theory are successfully applied to it. On other hand, these algorithms have become enough widespread due to their applicability for English language. In Russian language, the word order can be quite free what makes huge opportunities for poetry. Although, there is specific tradition of word order in a sentence. Taking this into account, it's obviously that methods of automated text analysis are successfully applied to texts with strict word order texts but not for poetic texts.

2. Preparation to reordering model building

2.1. Syntax constructions definition

According to free word order and some other facts we make a decision to equal a word order with prose texts, instead of development of a particular model for poetry. This decision allows us to use large text corpus to increase confidence for developed algorithms. On the other hand, it can be useful not only for poetry. A word order depends on syntax constructions called also syntax groups or chunks according to Noam Chomsky theory [5]. The splitting of texts to such constructions have been
demonstrated in CoNLL-2000 materials [6] which were adapted for Russian language. By its standards syntax group contains all vertex from left of chosen vertex which are in sub-tree including the chosen one. By other words, a chunk is a group vertex with its depended groups, excluding groups with vertex of same tag due to non-recursivity condition. In addition, a group contains the pre-modifiers like articles and adjectives. So, a primary case of group is non-nested one and all the rest cases have flat structure. The definition of hierarchy in groups is not an aim of research so let it be implicit. This usage of N. Chomsky theory contains some simplification of its clauses without any contradictions to theory. The tags used in CoNLL-2000 have obvious acronyms: NP – noun phrase, VP – verb phrase and etc, and we use almost all of them, except the ones like subordinate clause or list marker which are replaced with conjunction and numeric phrases correspondingly.

2.2. Text chunking

There are a lot of approaches based on neural networks for recognition of syntax groups. All of them have one sufficient disadvantage: the quality of successful predictions for minor classes is too low. We suggest a modification of this approach based on correction of ML algorithms by statistic classifier. Almost all popular ML algorithms, except support vector machine which requires at least five-times cross-validation, have an output parameter of probability. For classification tasks it looks like vector of probabilities for object belonging to each class. We can easily consider these probabilities as predictions confidence measure. The main idea of this paper is that the significant quantity of popular methods, like XGBoost or multilayer perceptron or any neural network based on them, can be mistaken on minor class instances and have probably the insufficient but existing dependency from the train set ordering. The statistic classifier is expected to negate these side-effects, but its usage as standalone method will be hardly better than any rule-based one. Its input requirements are the same with classifiers mentioned above, but without black-box structure. Instead of this implicit entity it contains the frequency-rated correspondings feature vectors with class labels. Our particular statistic classifier have an interface compatible with other classifiers from Python’s scikit-learn module, because the frequency can be easily converted to probability. Although the case of usage of two or more classifiers as in this case is suitable for voting strategies, we don’t use it for several reasons. The first is a majority victory that for case of two classifiers is rather rare. The second is that the increasing of classifiers pool by duplicates follows the victory of duplicated one and addition of another kind of classifier like logistic regression, follows voting classifier precision equality with a logistic regression decision by definition of voting. If the voting will consider an argmax, statistic classifier predictions will have minor weights. So we use two classifier’s union strategies: a stacking [7] and a choosing classifier due to probability threshold. This combination is also implemented with scikit-learn interface, that allows us to retrieve an optimal value for threshold parameter with tool GridSearchCV. It takes a classifier instance and parameter values for input and calculates optimal parameter by comparing cross-validations results. After this operations we have value of 0.86. In the table 1 there are results for XGBoost, multilayer perceptron (MLP) and logistic regression combineted by stacking or threshold.

| Combination method                      | Second classifier | Average | Max  | Min  |
|----------------------------------------|-------------------|---------|------|------|
| Stacking with statistic classifier     | XGBoost           | 0.9     | 0.91 | 0.89 |
|                                       | MLP               | 0.91    | 0.92 | 0.87 |
|                                       | Logistic regression | 0.88    | 0.89 | 0.87 |
| Threshold-based combination with statistic classifier | XGBoost           | 0.91    | 0.93 | 0.9  |
|                                       | MLP               | 0.92    | 0.94 | 0.9  |
|                                       | Logistic regression | 0.88    | 0.89 | 0.87 |
The authors’ hypothesis are proved by these results, as shown in table 1, the logistic regression with such superstructure isn’t better than standalone one.

3. Words reordering algorithm

3.1. Conditional Random Fields

The bag-of-words approach is well proved for the text classification tasks. This approach means that each word occurrence is more significant than word order. Otherwise, there are such tasks as named entity recognition or POS-tagging (part-of-speech tagging), where the word sequences are, in reverse, more significant. Conditional random fields (CRF) [8] are useful in current research because this approach is working with word sequences, not with separate ones. Formula (1) is a conditional probability formal description, with \( y \) as hidden state, in this particular case — part of speech, and \( x \) is observable variable equal with an entity or its context:

\[
p(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \eta_k f_k(y_t, y_{t-1}, x_t) \right\}.
\]  

(1)

Generally, we have two components:

1. Normalization: the right part of equation contains no probabilities explicitly but describes them. As far as an we get a probability value at output, we need a normalization. The normalization constant \( Z(x) \) is sum of all possible state sequences which is equal to 1.

2. Weights and features: this component can be examined as logistic regression with weights and corresponding features. The weights estimation is performed by maximum likelihood estimation and the features are defined by user.

Another entity recognition methods, for example, regular expressions or graph models like hidden Markov models or maximum entropy Markov model can be useful in object identity. But CRF is popular and easy, that makes it best candidate for named entity recognition. CRF model is non-oriented graph model, which takes word’s context into account. For example:

\[ [\text{NP} \, \text{Мой дядя самых честных правил, Когда не шутку}] \, [\text{O} \, \text{правил, Когда}] \, [\text{PP} \, \text{в} \, \text{шутку}] \, [\text{VP} \, \text{занемог}]. \]

In this example NP is an acronym for noun phrase, VP is for verb phrase and PP is for preposition. This is the particular case of sequence marking task. BIO notation has been used to represent chunks as tokens sequences with labels. According to this notation, NP syntax group is represented as the beginning of group (B-NP) and all the rest in group as inner part (I-NP). The tokens which are not part of syntax group are marked as O.

Next step is preparation of training and test set for feature extraction. These features vectors correspond to objects in dataset. It’s most important process for ML approaches, because it sufficiently depends from feature extraction function. The default step is extracting features from words in context of chosen one on \( t \)-th position:

- \( w[t-2], w[t-1], w[t], w[t+1], w[t+2] \);
- \( w[t-1]|w[t], w[t]|w[t+1] \);
- \( \text{pos}[t-2], \text{pos}[t-1], \text{pos}[t], \text{pos}[t+1], \text{pos}[t+2] \);
- \( \text{pos}[t-2]|\text{pos}[t-1], \text{pos}[t-1]|\text{pos}[t], \text{pos}[t]|\text{pos}[t+1], \text{pos}[t+1]|\text{pos}[t+2] \);
- \( \text{pos}[t-2]|\text{pos}[t-1]|\text{pos}[t], \text{pos}[t-1]|\text{pos}[t]|\text{pos}[t+1], \text{pos}[t]|\text{pos}[t+1]|\text{pos}[t+2] \).

Here \( w[t] \) and \( \text{pos}[t] \) are corresponds to word and part of speech for word on \( t \)-th position. These features express word characteristics according to its context, for example \( w[t-1] \) and \( \text{pos}[t+1] \). In this example, \( w[0] \) means word and \( \text{pos}[0]|\text{pos}[1]|\text{pos}[2] = \text{DT}|\text{JJ}|\text{NN} \) means part of speech for it and next two words are DT and JJ correspondingly. CRF uses this data for training associations between features and labels for predicting labels sequence. CRFsuite is implementation of CRF, which expects any string value for feature containing no special characters as “;” or “=” and represents it as “name=value” for readability.
3.2. Words order model training

The easiest way to retrieve the labels is the training classifier for default chunking feature set. We trained it with about 10 000 sentences from dataset described above. The classifier was chosen because it can be submitted as classification task where objects are words in sentences and labels are BIO-tags. Chosen features are the next:

- word object morphology;
- morphology of word objects from left of chosen;
- morphology of word objects from right of chosen

We used context with window size 2, so we took the morphology for two tokens before and two after particular one. Using Python’s module to work with CRFsuite [8], next feature templates were chosen:

- \( w[t - 2], w[t - 1], w[t], w[t + 1], w[t + 2] \), lexical information;
- probability of next token after previous (context \( \pm 1 \)): \( w[t - 1]w[t]w[t + 1] \),
- grammar information (context \( \pm 2 \)): \( pos[t - 2], pos[t - 1], pos[t], pos[t + 1], pos[t + 2] \),
- probability features (context \( \pm 2 \), probabilities of tag after previous tags): \( pos[t - 1]pos[t], pos[t]pos[t + 1], pos[t + 1]pos[t + 2], pos[t + 2]pos[t - 2]pos[t - 1]pos[t], pos[t - 1]pos[t]pos[t + 1], pos[t]pos[t + 1]pos[t + 2] \).

The training set was automatically converted to special CRFsuite format, which was used for feature extraction and then we’ve applied the logistic regression to feature matrix. It shows 87% of precision on test set.

For chunking task it’s correct to use the precision and the recall as a quality measure. Here we mean that precision is a percent of right found groups from all at the output. The recall is a percent of right found chunks in train set. Both of them takes into account a relevance of whole group, not only for its separate tags.

3.2.1 Word ordering algorithm. The next algorithm performs words order alignment or, in other words, leads it to the modern Russian orthography. Something similar was made in [9], but for other purposes, in particular, for the text information indexing. Also, the method from [9] unlike from the defined below didn’t takes syntax parsing, and the words order problem wasn’t actual for the kind of texts examined there. We suggest an algorithm based on text chunking as on pre-processing and we use its results as input of recurrent neural network such as LSTM or CRFsuite [8]. Last one hasn’t neural network topology, but it could act the same way, due to Viterbi algorithm in its base which is working with hidden Markov models. These models have the same context memory evolving during training process as the recurrent neural networks. The algorithm demands next preparation steps:

1. Syntax chunker, that we describe above, training on modified corpus of SynTagRus [10] which contains also a different variants of word order, equivalent to unordered n-grams used in [11] so the word order entropy is taken into account by synthetic data addition.
2. By the pymorphy2 [12] module usage, the morphological analysis is performing and corresponds each word with its syntax features and sends it to input of recurrent neural network.

The algorithm:

1. Recurrent neural network acts like the classifier which corresponds each syntax group or chunk with offset value from -2 to 2. Zero value means that a group is on suitable position.
2. According to obtained probabilities, the syntax groups are dynamically replaced.
3. Another one neural network with the same topology also classifies the words in chunks by its offsets, according to morphological features.
4. According to obtained probabilities, the words in each syntax groups are dynamically replaced.

The algorithm implemented by Python language.

3.2.2 An example. As the example we’ve chose the first line from A. Pushkin poem “To Natasha”:

1. After chunker’s markup it looks: [VP вянет, вянет] [NP лето красно];
2. After CRFsuite or LSTM work it is: [NP лето красно] [VP вянет, вянет] because it’s obvious that in Russian a verbal group should be after noun one, and model has trained this.

3. Then words in groups were replaced like: [NP красно лето] [VP вянет, вянет].

The method usage is a splitting text to chunks with syntax groups by training CRFsuite model on SynTagRus corpus and applying it to text. For further processing a user obtains text with straight or grammar word order which follows the several new opportunities. The first is a text adaptation to word2vec-based tools usage and the second is a retrieval of new features. The example of these features are the synonyms obtained from a cosine distance or meaningful words or phrases based on words mover distance which can expand features for classification approach from [11] that allows to reach better results at classification by style or genre.

| Table 2. Classifier’s results before and after words reordering |
|---------------------------------------------------------------|
| Classification without words order operations | Average 0.71 | Max 0.74 | Min 0.67 |
| Classification after words order operations | Average 0.83 | Max 0.86 | Min 0.82 |
| Classification after words order operations with pre-training | Average 0.9 | Max 0.93 | Min 0.87 |

4. Conclusion

In this paper we suggest an approach for work with poetry texts with non-strict word order. This approach is bringing a text to grammar word order by recurrent ML algorithms based on syntax chunking. In addition, the application of this algorithm allows us to use a large corpus of non-poetic text for model training with no contradictions to dataset homogeneity. It can be also useful not only for the analysis of poetry but for news articles or blogs too [13].

Acknowledgments

The work was supported by Russian Science Foundation grant № 19-18-00466.

References

[1] Van Kemenade A 2002 Word order in Old English prose and poetry: The position of finite verb and adverbs Topics in English Linguistics 39 pp 355–372
[2] Evseev V Ya 1948 Runes of “Kalevala” and the Russian-Karelian folklore communication Bull. Kar.–Fin. Sci. Base of Ac. Sci. of the SU pp 69–89 (in Russ.)
[3] Filatova O M 2007 Linguo-aesthetics components as an object of translation activity (based on the material of the poem by H. Heine about Lorelei) Bull. of the Udmurt University. Series «History and Philology» pp 193–203 (in Russ.)
[4] Roscow G H 1981 Syntax and style in Chaucer’s poetry vol 6 (Boydell & Brewer Ltd.)
[5] Chomsky N 2014 The minimalist program (MIT press)
[6] Sang E F and Buchholz S 2000 Introduction to the CoNLL-2000 shared task: Chunking. (arXiv preprint cs/0009008)
[7] Wolpert D H 1992 Stacked generalization Neural networks vol 5 no 2 pp 241–259
[8] Kidinov M S 2013 Shallow Parsing of Russian Text with Conditional Random Fields Machine Learning and Data Analysis vol 1 no 6 pp 714–724 (in Russ.)
[9] Barakhnin V B and Kupershtokh A A 2006 The coordinate indexing algorithm of electronic scientific documents Computing and information technology in science, technology and education technology and education 1 pp 228–232 (in Russ.)
[10] Dyachenko P V, Podlesskaya O Yu and Sizov V G 2014 NKRYA: main body and SintagRus, parsing of texts with removed morphological homonymy Information Technologies and Systems (ITiS’2014) Collection of works pp 150–156 (in Russ.)
[11] Barakhnin V, Kozhemyakin O and Pastushkov I 2017 Automated determination of the type of genre and stylistic coloring of Russian texts *ITM Web of Conferences* 10 https://doi.org/10.1051/itmconf/20171002001

[12] Korobov M 2015 Morphological analyzer and generator for Russian and Ukrainian languages *International Conference on Analysis of Images, Social Networks and Texts* pp 320–332

[13] Barakhnin V B, Kuchin Ya I and Muhamedyev R I 2018 On the problem of identifying fake news and algorithms for monitoring them *Materials of the III International Scientific Conference Informatics and Applied Mathematics* 1 pp 213–218 (in Russ.)