Unsupervised Tokenization Learning

Anton Kolonin
Aigents
akolonin@gmail.com

Abstract

In the presented study, we discover that so called "transition freedom" metric appears superior for unsupervised tokenization purposes, compared to statistical metrics such as mutual information and conditional probability, providing F-measure scores in range from 0.71 to 1.0 across explored corpora. We find that different languages require different derivatives of that metric (such as variance and "peak values") for successful tokenization. Larger training corpora does not necessarily effect in better tokenization quality, while compacting the models eliminating statistically weak evidence tends to improve performance. Proposed unsupervised tokenization technique provides quality better or comparable to lexicon-based one, depending on the language.

1. Introduction

The language modeling based on unannotated corpora as unsupervised language learning have been attracting great attention over last years and getting significant results with transformer-based models such as BERT and GPT relying on deep learning neural networks (DNN) [1, 2]. At the same time, the concept of unsupervised learning for a language grammar represented in an "interpretable" representation with formal grammar such as Link Grammar has been suggested by Goertzel and Vepstas in 2014 [3]. Another approach for the problem has been posed by Kolonin coming up with concept of so-called "deep patterns" with hierarchical "symbolic" grammatical pattern structures learned from the texts as a way to model grammars and ontologies for natural languages suiting wide range of practical applications [4]. Further studies on this path performed by Glushchenko and colleagues have indicated the possibility to learn the grammars as well as domain ontologies given the high-quality parse trees of the texts obtained from unannotated training corpora [5, 6]. Unfortunately, the critical part of the pipelines described in the latter studies has turned to be the unsupervised generation of the parses which has turned to be low quality being based on simple "minimum spanning tree" - either based on mutual information (MI) [7] or "contextual information" [6] extracted from the BERT-based "deep learning" model [1]. Still, further studies made by Ramesh and Kolonin have shown possibilities of building different natural language processing (NLP) applications based on a language model represented by formal grammar (Link Grammar in the explored cases) [8, 9, 10]. Along the way, in the latter works, the concept of "interpretable natural language language processing" (INLP) has been introduced to indicate the domain of NLP explorations involving both learning language models represented in an interpretable form and applying these models for different applications such as text segmentation, language generation and question answering.

Yet another problem which has its place in case of conventional language model learning based on DNN [1, 2] as well as in case of "interpretable" unsupervised language learning advocated in [4, 5, 6] is tokenization. In most of the cases tokenization is based on predetermined rules and dictionaries, which does not quite fit the "grand plan" of completely unsupervised language learning from the scratch with no any prior knowledge of the language, including the knowledge regarding the lexicon and punctuation [3]. Thus, objective of the presented study was to evaluate the possibility of learning the sets of tokens representing both punctuation and lexicon – without of any prior knowledge about the language at all, so the set of valid combinations of letters or characters specific to punctuation marks or valid lexical entities such as words for a language is learned along with the tokenization process.

The starting points were found in works of Kearsley [11] and Wrenn with colleagues [12], who tried to explore possibility of unsupervised segmentation applied to different languages [11] and domain-specific literature [12].

The former work [11] provides exhaustive overview of different tokenization techniques appleid to different languages, exploring different methods and metrics. Unfortunately, the F1 scores reported in this work for completely unsu-

pervised tokenization based on statistical measures appear not high enough, so further we follow this approach trying to outperform these scores on the set languages relevant and available to us, focusing on unsupervised tokenization only.

The latter work [12] focuses on unsupervised tokenization based on statistical measures such as conditional probability (CP) as well as introduces so-called "freedom of transition" (we further call it "transition freedom" or TF) metric, which appears fundamentally consistent with notion of "free energy" suggested by Friston [13] as a key for an artificial intelligence concept. The TF in context of [12] corresponds to the number of symbolic states (characters, letters or N-grams) that can be following after the current state or preceding the current state. Then, the sharp increase of the TF level along the temporal sequence of states might correspond to the loss of the Friston's "equilibrium" [13] and so the "tokens" might be considered as chains of state resting in conditions of mutual equilibrium framed with transitions with loss of this equilibrium marked with the TF level bursts. Further we explore both statistical measures and TF metrics finding the latter substantially more practical. In particular, we explore different metrics based on the CP and TF such as derivative, variance and "peak values" introduced in [12] indicating the expressed local maximums on the derivative curve along the text being tokenized.

Interestingly, Kearsley writes that "Given that the human ability to successfully read any natural language provides an existence proof that a generalized segmentation system (as implemented in the human mind) is possible, it is reasonable to investigate the feasibility of a language-agnostic segmentation system that could be easily integrated into larger natural language processing systems" [11]. Extending this statement, we anticipate that advance in this area could be also beneficial to deal with any sequential data such as flows of events and states in experiential or reinforcement learning. In particular, the "global feedback" concept suggested in work [14] demonstrates good learning rates in case when the cognitive schema leading to the feedback or reward can be reliably associated with entire sequence of preceding actions which is difficult in existing reinforcement learning frameworks. The ability to segment sequences of cognitive experiences unsupervisedly might potentially advance research on reinforcement and experiential learning with delayed reward or with no explicit feedback in any subject domain beyond the NLP. The importance of the latter goal is also outlined by Schmidhuber and colleagues saying that "discovery of reusable sub-routines simplifies decision-making and planning in complex reinforcement learning problems" [15].

As it will be presented further, we find the TF to be superior over Mutual Information (MI) [5, 6, 7, 11] and CP [12] for unsupervised text segmentation (tokenization) task. We find that English and Russian languages require one specific way (variance) of handling the TF while Chinese requires a bit different specific way (derivative-based "peak values") for the same purpose. Tokenization quality for English and Russian may have F1 scores as high as F1=0.96-1.0, depending on training and testing corpora while for Chinese the best score is F1=0.71 with precision of word discovery in lexicon reaching 0.92. Larger training corpora does not necessarily effect in better tokenization quality, while compacting the models eliminating statistically weak evidence typically improves the quality. Unsupervised TF-based tokenization provides quality same or better than lexicon-based one for English and Russian while for Chinese it appears to be the opposite (as it could be anticipated), while the precision of the lexicon discovery for Chinese using it appears close to reference tokenization. Doing English and Russian tokenization with removed spaces makes the situation similar to Chinese, where use of whitespaces is not common, with reasonable quality on lexicon-based tokenization but much worse results on TF-based one.

2. Data Sets

We have used different training data sets for three different languages such as English, Russian and Simplified Chinese (Mainland Chinese) while the same parallel corpus have been used for testing.

For Chinese train corpora we have used CLUE benchmark News 2016 dataset obtained under https://github.com/brightmart/nlp_chinese_corpus link. The dataset has two pieces – Train and Validation, each of the pieces was used as individual training dataset. The raw data encoded in JSON have been processed so that "title", "desc" and "content" fields have been extracted individually and each of the three fields was saved on separate line in the text file used as input for further pro-

cessing. After such processing, we have got 270M size of Validation dataset and 8,500M of Train dataset.

For English train corpora we have used Brown (6M size) available at http://www.sls.hawaii.edu/bley-vroman/brown_nolines.txt, Gutenberg Children (29M size) and Gutenberg Adult (140M size) collections from https://www.gutenberg.org/ as well as mixed collections such as Gutenberg Children and Adult blended together and, finally, all three Corpora blended together.

For Russian train corpora we have used RusAge collection from https://www.kaggle.com/datasets/oldaandozerskaya/fiction-corpus-for-age-based-text-classification, as two separate pieces Test (141M size) and Previews (825M size).

For test corpus across all three languages above we have used Parallel Chinese/English/Russian corpus of 100 multi-sentence statements on financial domain derived from the one available at https://magichub.com/datasets/chinese-english-parallel-corpus-finance/. The original corpus is parallel Chinese/English, but Russian version of all 100 statements have been added with help of Google Translate service, having the Chinese proper names manually renamed to Russian or English proper names used in appropriate subject domain context.

Moreover, the reference lexicons have been been obtained. English lexicon have been obtained from https://raw.githubusercontent.com/aigents/aigents-java/master/lexicon_english.txt, Russian lexicon have been obtained from https://raw.githubusercontent.com/aigents/aigents-java/master/lexicon_russian.txt.

Few different Chinese lexicons were obtained for reference: CLD at http://www.chineselexicaldatabase.com/download.php [16], BCC (BLCU Chinese Corpus) at https://www.plecoforums.com/threads/word-frequency-list-based-on-a-15-billion-character-corpus-bcc-blcu-chinese-corpus.5859/, and SUBTLEX-CH at http://crr.ugent.be/programs-data/subtitle-frequencies/subtlex-ch.

3. Exploration Methodology

3.1. Methodology Overview

The methodology of the study have involved few different phases applied to all corpora corresponding to all three languages.

First, the models have been built for every train corpora across all three languages.

Second, tokenization have been performed for each of the languages, using the same parallel test corpus, relying on the models created on the previous phase, using different training corpora with different metrics and hyper-parameters as will be discussed further. While doing the tokenization, F-score (F1) was evaluated for every set of hyper-parameters and selected metrics, comparing the tokenization outputs with outputs of reference "standard" tokenizer. At this point, the corpora and sets of hyper-parameters leading to the best F1 scores per language have been identified.

Third, the tokenization configurations corresponding to winning F1 scores have been evaluated compared to reference lexicon-based tokenizer, specific for each of three languages.

Forth, the winning configurations have been evaluated for precision of lexicon discovery, making sure which fraction of the tokens identified by the best unsupervised tokenizer setup actually corresponds to entries in reference lexicon dictionaries for each of the languages.

3.2. Model Structure and Building

Each of the models created for a corpus has been represented by three pieces, based on N-grams with N in range from 1 (unigrams or individual characters or letters) to N_{max} (up to 7, according to discussion in [12]), with the latter being one of the two hyper-parameters discussed further. These pieces are described below.

- N-gram frequencies or counts of N-grams experienced through the corpus.
- Counts of all N-grams appearing after every specific N-gram (call them "forward transitions").
- Counts of all N-grams appearing before every specific N-gram (call them "backward transitions").

The model building process have been applied to corpus data on line-by line basis, according to the original corpora text layout, without of any other pre-processing.

For transition counts, two different models have been built for every language corpus. First, there was N-gram-to-symbol counts where number of single symbols (unigrams) following or

preceding every possible N-gram have been counted. Second, there was N-gram-to-N-gram transitions, where following and preceding N-grams were counted in respect to the other N-gram, having the N the same. Preliminary studies on English corpora run at the beginning of our exploration have shown worse performance of the latter kind of models, so further studies were involving the N-gram-to-symbol models only.

The N rank of N-gram was varying from 1 to 7 except Chinese, with N_{max}=3 for smaller Validation dataset and N_{max}=2 for larger Training dataset, due to memory restrictions of 32G RAM which made impossible to process larger models for Chinese corpora.

The described model of a language based on given corpora can be represented as a bi-directed graph, with transitions on graph edges pointing both forward and backward independently, with the every symbolic unit involved in multiple over-laid subgraphs due to multiple contexts represented by embedding the same N-gram in multiple transitions on the graph as well as due to the embedding of the N-grams with lower rank into multiple N-grams with higher rank. The bi-directed graph is weighted by frequency counts associated with the vertices, corresponding to Ngrams as well as with the edges, corresponding to transitions. The same graph might be viewed in three ways. First, it can be thought as an excessive container including a graph-based grammatical model expressed in a formal grammar such as Link Grammar [3, 5, 6]. Second, it can be considered as a bottom layer of the heterarchical system of "deep patterns" which can be used to infer higher-level abstractions [4]. Third, it can be seen as a set of interconnected symbolic "instances" underlying the abstract higher level language model consisting of interconnected symbolic "invariants" with those invariants corresponding to parts of speech, in terms of Vityaev and colleagues [17].

3.3. Tokenization Methods and Metrics

The following tokenization methods and respective metrics were used.

- Greedy aggregation of symbols into tokens according to the mutual information (MI) computed for the pairwise symbol associations like it is described in earlier works [7, 11]. This did not work well in the initial cursory study on English corpora, systematically breaking the proper English words to pieces so this was not systematically studied further as a promising approach to tokenization.
- Probability (P) of an N-gram – selection of N-grams with lexicon-wise probabilities above the thresholds as a delimiters, breaking the stream of symbols into tokens.
- Conditional Probability (CP) computed on N-gram-to-N-gram transitions derivative in both forward and backward directions, as described in earlier works [11, 12], with local maximums on N-gram-to-N-gram transitions corresponding to the token breaking points.
- CP variance – based on the above as a difference between the CP and its mean value for given input sequence.
- Transition Freedom (TF) as number of possible transitions on forward or backward model graph traversal at specific N-gram according to methodology described by our predecessor [12], with values exceeding the threshold breaking the stream of symbols into tokens.
- TP variance – based on the above as a difference between the TF and its mean value for given input sequence.
- TF derivative in both directions, with local maximums on N-gram-to-N-gram transitions corresponding to the token breaking points.
- TF "peak values" defined in [12] as a value of TF derivative on previous transition minus value of TF derivative on following transition, which outlines sharp positive extremums of the TF curve along the processed sequence of N-grams, indicating the token boundaries.
- Lexicon-based tokenization in "greedy" mode, so that either longest or most frequent token entry present in the language-specific lexicon dictionary is identified as a next token when traversing the input text forward from left to right (also, the option has been explored to blend the two criteria of the length and frequency logarithm are blended). As this is is not an unsupervised

approach, being based on pre-created lexicon, this tokenization was used only for reference.

- Reference "hardcoded' tokenizer used to assess the F1 score of the unsupervised tokenization. In case of English and Russian it was simple text split based on white spaces with quotes, brackets, periods, commas, semicolons and other punctuation symbols detached from the split token sequence. In case of Chinese it was Jieba Tokenizer which is using combination for hardcoded rules, built-in dictionaries and probabilistic measures https://github.com/fxsjy/jieba [18].

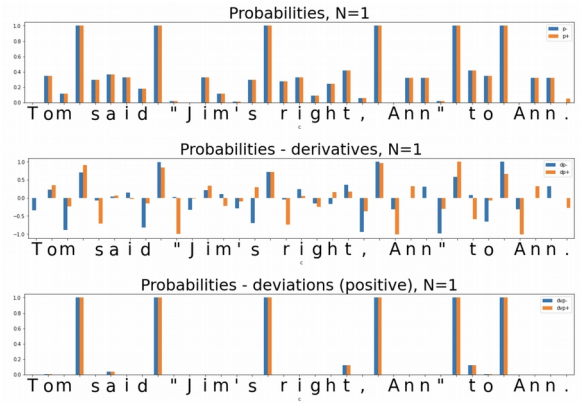

Figure 1: Using probabilities (p) and derived metrics such as variance (dvp) and derivates in forward (dp+) and backward (dp-) traversals. It is clearly seen that quotation and punctuation marks can not be isolated from the words.

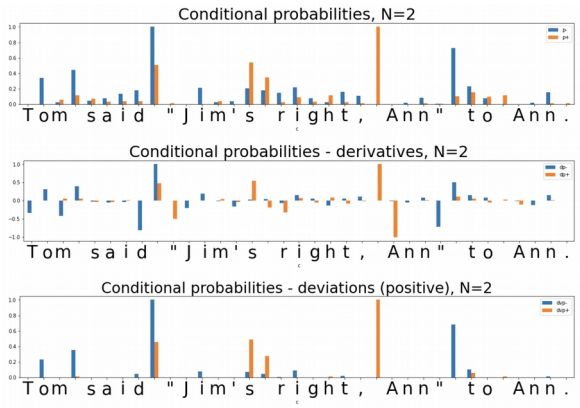

Figure 2: Using condition probabilities (p) and derived metrics such as derivate in forward (dp+) and backward (dp-) transitions and variance (dvp+ and dvp-, respectively) computed on bigrams. It is clearly seen that quotation and punctuation marks can not be isolated from the words while some of the words might get disassembled into pieces.

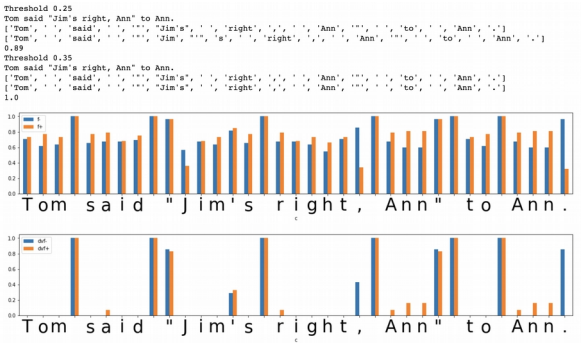

Figure 3: Using transition freedoms in forward (f+) and backward (f-) directions and their variances (dvf+ and dvf-) computed on unigrams being able to identify all words as well as punctuation marks clearly, with threshold values of 0.25 and 0.35.

For all methods relying on CP and TF metrics above, two alternative ways of identifying the token boundaries are possible. First, as suggested in [12], the "mean" metric is computed on forward and backward traversal metrics over the sequence of N-grams referring to corresponding subgraph in the model. Second, the token break is identified as a metric derived from either P, CP or TF exceeding the threshold on either forward or backward transition along the text, so the "max" was used instead of the "mean". While the former "mean" method is described in [12], the cursory checks across corpora has shown it it is not quite reliable compared to the following alternative so the latter "max" method was used in the studies presented below.

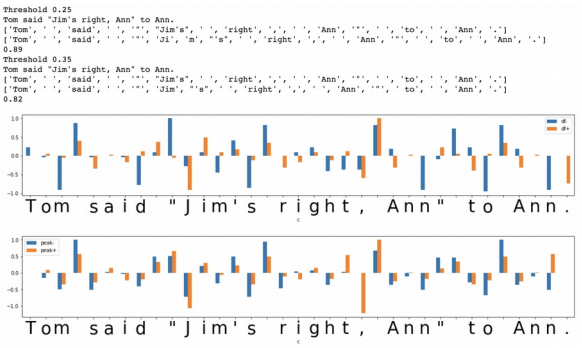

Figure 4: Using transition freedom derivatives in forward (df+) and backward (df-) directions and their "peak values" (peak+ and peak-) computed on unigrams being not able to identify all words clearly and failing to separate punctuation marks, with threshold values of 0.25 and 0.35.

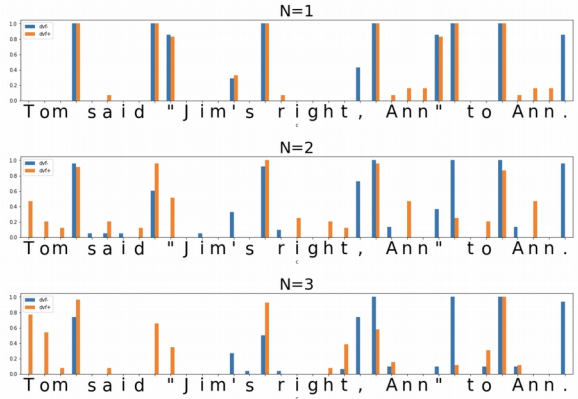

Figure 5: Variances of transition freedoms in forward (f+) and backward (f-) directions with different N-gram ranks N, clearly showing that N=1 appears to be the best to identify whitespaces as well as punctuation marks.

It was explored if either of the P/CP-based tokenization metrics or TF-based one can be referred as a token break indicator across different values of N rank of N-grams, picking either as a mean ("mean") or a maximum ("max") across all values of N selected as hyper-parameter. The cursory checks has shown that the latter option with selection of "max" does not perform better than the former "mean" version so the mean across different N values was used in studies presented further.

While trying to reach the superior F1 scores during the studies, we have also explored if it would help to "compress" the model eliminating the edges on the graph with weights below certain threshold measured relatively to maximum N-gram or transition frequency in the local subgraph segment. That is, it was tried if a model derived from the raw model with removal of all low-frequency N-grams across all N-grams, and removal of low-frequency transitions for any given N-gram can increase the F1 performs better.

3.4. Tokenization F1 score and precision of lexicon discovery

The evaluation of F1 score has being performed to compare performance of the unsupervised tokenization against reference tokenization based on "hardcoded" logic. It has been computed based on non-unique set of tokens with counted occurrences, so that, for example, each repetition of determiner "the" in a tokenized text is considered separately.

The other evaluation of quality of unsupervised tokenization was considered to be its capacity to discover lexical entities for an unknown language, called precision of lexicon discovery evaluated as ratio of all tokens found in an input text present in a reference lexicon dictionary denominated by total number of tokenized entries.

3.5. Tokenization Hyper-parameters

According to the discussion above, there were just few hyper-parameters explored in further experiments on unsupervised tokenization in a unified way across all three languages studied.

- Tokenization metric – use either P/CP or TF as a base metric and then use either base metric value on itself or use a derived version of it such as variance or derivate or "peak value".
- Combination of N ranks used to do the model graph traversal and "mean" metric computation based on specified subset of N-grams only. We have explored options with using every possible N individually as well as arbitrary combinations of N values.
- Model compression threshold used to remove low-frequent N-grams vertices and transitions between them on the model graph. We have used values 0.0 (corresponding to no compression at all), 0.0001, 0.001, 0.01, and 0.1.
- Tokenization metric threshold identifying the level so the value of a metric exceeding this level would correspond to the token boundary. We have used values 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9.

We were using "grid search" to find the best configuration for these four hyper-parameters in order to identify the most well performing setup, providing the best F1 score.

The "winning" configuration of the hyper-parameters obtained for the full test set of 100 sentences for a language was independently verified for independent splits of this set in the two sets of 50 sentences obtaining nearly the same results for the same combinations of the hyper-parameters without of change of the "winning" configuration.

3.6. Symbol Category Clustering

We have tried to explore to which extent the transition-based model can be used to identify cate-

gories of different symbols. For this purpose, we have performed agglomerative clustering of the symbols into dendrogram tree based on the similarity of symbols (as N-grams with N=1) stored in the model in the vector space of their adjacent transitions both in forward and backward transitions, based on Cosine and Jaccard similarity measures.

3.7. Space-less ("fluent") Text Segmentation

We have also tried to run the same text tokenization experiment on the input English and Russian texts having white spaces removed, to understand the limits of the approach to tokenize continuous ("fluent") text or speech with no regular and explicit punctuation, like it takes place in Chinese.

4. Experimental Results

4.1. English

The maximum performance of tokenization that we were able to achieve was F1=0.99, obtained with TF variance metric based on training on smallest Brown corpus, N=1 (unigrams), model compression thresholds 0.0001 and 0.001, and tokenization threshold 0.4 and 0.5 – respectively. Any larger corpora or combining the corpora were making it possible to reach F1 above 0.93 but below 0.99 with similar hyper-parameters.

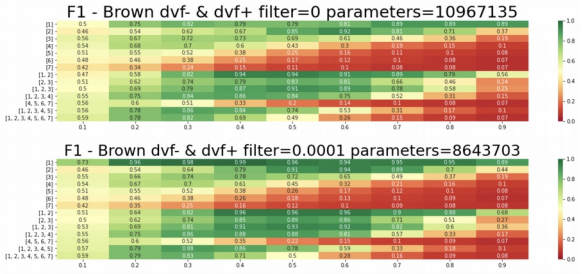

Figure 6: Heat-maps rendering F1 scores obtained with unsupervised tokenization for training on English Brown corpus with no model compression (top) and model compression with 0.0001 threshold (bottom) with different combinations of N (vertical axes) and different tokenization thresholds (horizontal axes). It is seen that the highest F1 scores above 0.96 correspond to model compressed with threshold 0.0001, N=1 (unigrams) and thresholds 0.3-0.4. Each of the models (non-compressed at the top and compressed at the bottom) have numbers of model parameters indicated in the plot titles, where each parameter corresponds to weight or frequency count for either N-gram or transition between N-grams.

Using lexicon-based tokenization in "greedy" mode, driven by length of the token provided the same level of performance with F1=0.99, after having delimiting symbols added to the reference lexicon dictionary.

Unsupervised tokenization on spaceless ("fluent") text has shown to have F1=0.42, while the Lexicon-based on the same provided F1=0.79 (comparable fo Chinese F1=0.82 mentioned further), if obtained with search driven by weight as multiplication of token length and logarithm of token frequency. Apparently, such results can be explained by lack of word stress articulation and speech pauses in spoken communications. In case of dictionary-based tokenization, expectedly, it can be improved based on concurrent construction of alternative tokenization tree maximizing the weight across entire tree, like it is being done in Link Grammar and MST Parser [3, 5, 6] in case of phrase structure parsing at sentence level.

Precision of word discovery with unsupervised tokenization has turned to be 0.99 (after correction for proper English words missed in the reference lexicon dictionary), comparable with reference delimiter-based tokenization (1.0). The 0.01 error fraction was caused primarily by inability of the unsupervised tokenizer to recognize the question marks being attached to the ends of words as part of the tokens, which expectedly might be solved with larger corpora, involving greater variety of question mark experiences in different contexts, because all other punctuation marks have been identified as separate tokens correctly.

4.2. Russian

The maximum performance of tokenization that we were able to achieve was F1=1.0, obtained with TF variance metric based on training on any used corpora, N=1 (unigrams), model compression thresholds 0.0001 for all training corpora and even with no compression at all for smaller corpus, with tokenization threshold 0.7.

Using lexicon-based tokenization in "greedy" mode, driving the token search by length of the token, provided the lower level of performance with F1=0.94, due to the words missed in the lexicon, after having delimiting symbols added to the reference lexicon dictionary.

Unsupervised tokenization on spaceless ("fluent") text has shown to have F1=0.26, while the Lexicon-based tokenization on the same provided F1=0.72, if obtained with search driven by weight as multiplication of token length and logarithm of

token frequency. The same comments as in case of English language above applies here.

Precision of word discovery with unsupervised tokenization has turned to be 1.0 (after correction for proper Russian words missed in the reference lexicon dictionary), comparable with reference delimiter-based tokenization (1.0).

4.3. Chinese

The maximum performance of tokenization that we were able to achieve was F1=0.71, obtained with TF "peak" metric based on N=2 (bigrams) and model compression threshold 0.001 on larger train corpus with any tokenization threshold from 0.05 and below down to 0.0. Unfortunately, we were not able to explore the N-grams with N higher than 3 for smaller lexicon and 2 for larger lexicon due to 32G memory limit applied to our model implemented in Python using plain dictionaries of dictionaries design for the graph model storage.

Using lexicon-based tokenization in "greedy" mode, relying on the token search driven by length of the token, provided the higher level of performance with F1=0.83, due to the words missed in the lexicon.

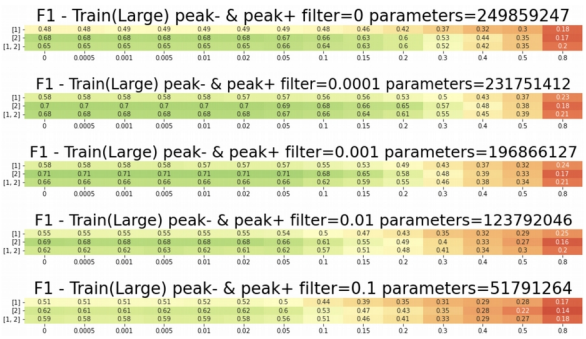

Figure 7: Heat-maps rendering F1 scores obtained with unsupervised tokenization for training on Chinese CLUE benchmark News 2016 corpus with no model compression (top) and model compression with different thresholds from 0.0001 to 0.1 (top to down) with different combinations of N (vertical axes) and different tokenization thresholds (horizontal axes). It is seen that the highest F1 scores at 0.71 correspond to model compressed with threshold 0.001, N=2 (bigrams) and thresholds from 0.05 and down to 0.0.

Apparently, in both cases above, the mistakes in tokenization referring to Jieba tokenizer did not have significant impact on the meaning as long as translation of alternative combinations of symbols could be looked up in Google translate given the lack of real knowledge of Chinese language by the authors of this research.

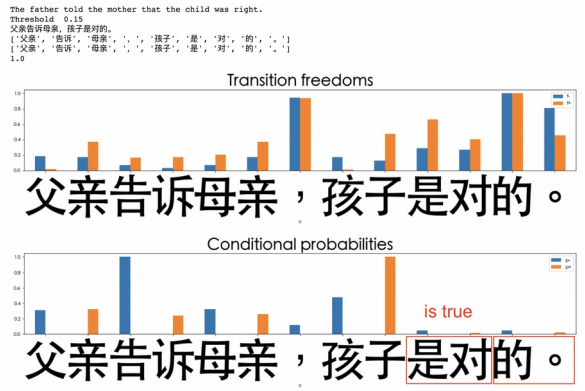

Figure 8: Using transition freedoms in forward (f+) and backward (f-) directions computed on bigrams for Chinese being able to identify all proper words as well as punctuation marks clearly, with threshold value of 0.15 (on the top chart). Use of probabilities (p+ and p- on the bottom chart) computed on bigrams fail to identify Chinese period "。" separately (rightmost bounding box at the bottom) as well as separate two symbols as individual tokens, while the word having these two symbols together appear still having valid sense in context of the input sentence translated as "is true" literally.

One of the typical tokenization errors has turned to be inability to detach Chinese-style period ("。"), like it was found in respect to question mark in English. Apparently, this could be solved using richer training corpus with greater diversity of contexts where the punctuation is used.

Precision of word discovery of Freedom-peak-based tokenizer is at least 0.92 (after correction for major omissions in the reference lexicon dictionary), comparable with one based on reference Jieba tokenizer (0.94).

4.4. Symbol Category Clustering

The symbol category clustering run on both English and Russian languages have shown general ability to identify proper groups of English and Russian symbols and letters as well as universal language-agnostic punctuation marks. It is interesting that even using Russian corpus, it was possible to obtain proper categories of English letters. It is even more interesting, that the symbol category trees for English language obtained relying on Russian corpus had more clean separation of vowels and consonants into individual tree branches, as well as punctuation marks (like opening and closing brackets and quotes), digits, et. cetera. It can be probably explained by greater "cleanness" of the English texts embedded in the Russian texts – as we remember, the best unsupervised tokenization results for English reported

above were obtained on smallest Brown corpus. Using both Cosine and Jaccard similarity measure were delivering similar results while the categorial trees based on Jaccard measure appear more well-balanced and reliable.

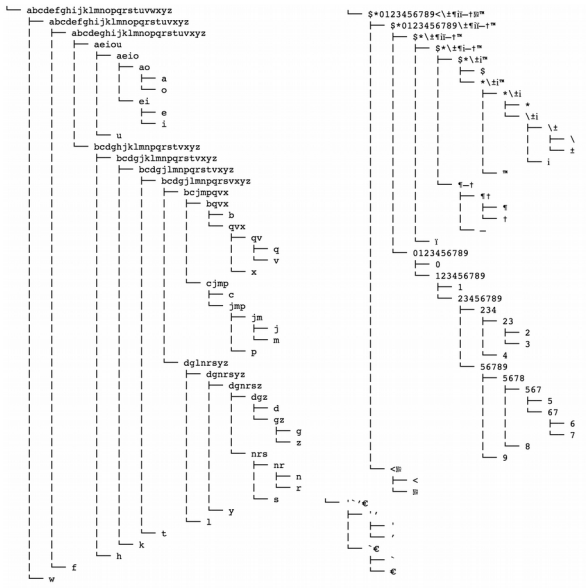

Figure 9: Symbol category agglomerative clustering trees using Jaccard similarity based on RusAge corpus identifying vowels, consonants, digits as well as punctuation mark groups.

5. Conclusion

Unsupervised tokenization based on Transition Freedom (TF) F1 scores, for English and Russian – especially, appears good enough as initial approximation for further applications of self-reinforcement learning as part of interpretable unsupervised learning of natural language.

New state-of-the art (SOTA) baseline for unsupervised tokenization has been introduced. The baseline may be further extended increasing complexity and richness of the test texts.

Optimal thresholds and specific TF-based metrics are specific to language. The process and policy of their discovery and adjustment in an unsupervised way should be further explored.

Clustering or parts of speech on space of transition graphs may provide some insights on morphology and punctuation structure of low-resource and domain-specific languages.

Hybridization of TF-based tokenization approach with lexicon-based one might be efficient for low-resource and domain-specific languages.

Further unsupervised grammar learning experiments, advancing the earlier studies [5, 6] can be run on the basis of suggested unsupervised tokenization framework.

Language	Tokenizer	Tokenization F1	Lexicon Discovery Precision
English	Freedom-based	**0.99**	**0.99** (vs 1.0)
English	Lexicon-based	0.99	-
English no spaces	Freedom-based	0.42	-
English no spaces	Lexicon-based	0.79	-
Russian	Freedom-based	**1.0**	**1.0** (vs 1.0)
Russian	Lexicon-based	0.94	-
Russian no spaces	Freedom-based	0.26	-
Russian no spaces	Lexicon-based	0.72	-
Chinese	Freedom-based	**0.71**	**0.92** (vs 0.94)
Chinese	Lexicon-based	0.83	-

Figure 10: Summary of the presented research on Unsupervised and Dictionary-based tokenizations relying on Transition Freedom (Freedom-based in the table), referring to rule-based or hybrid tokenizers (the last column provides reference numbers for rule-based/hybrid tokenizer in parentheses). English: both Tokenization and Lexicon Discovery tasks are solved with Freedom-based tokenization no worse than with Lexicon-based one (F1 and Precision at 0.99). Russian: both Tokenization and Lexicon Discovery tasks are solved better (F1 and Precision at 1.0) with Freedom-based tokenizer than with Lexicon-based one (F1=0.94). Chinese: Tokenization task is solved by Freedom-based tokenizer worse than by Lexicon-based one yet comparable to it (0.71 vs. 0.83), however Freedom-based tokenizer performs pretty well compared to rule-based/hybrid tokenizer (0.92 vs 0.94).

Applications for other Experiential Learning (EL) environments such as [14], including the ones with delayed/sparse reward, may be explored using the TF-based segmentation to identify natural boundaries of states and actions for application of the global feedback as well as using Reinforcement Learning (RL) techniques with self-reinforcement on historical data within unsupervised learning framework.

Limitations

The following limitations are know and have to be considered when applying the results of this work or relying them in the derived studies.

- It is well understood that in some cases tokenization based on N=1, according to the hyperparameters appeared to be the best for English and Russian, will not work, taking points inside decimal numbers and Web addresses as examples. It might be improved with N>1, but given slightly worse performance of such setup agains the explored test

set needs further study. Potentially, notion of the broad tokenization context (attention in terms of [1]) has to be introduced to improve the suggested technology to scale up dealing with more rich test corpora.

- The test corpus of 100 sentences appears covering quite limited subject domain (personal finance) being not sufficiently rich, so evaluation on the larger and richer corpora is recommended for further studies and applications.

- While the use of unsupervised parsing based on Mutual Information (MI) [3,5,6,7] have been cursory checked and considered not practical, full scale evaluation of it has not been performed so no firm claim of its futility can be made, it might be explored and verified further.

- While use of "peak values" have been explored for Transition Freedom (TF) as suggested in [12], no similar "peak values" have been systematically tried fried for Conditional Probability (CP) [11,12]. Cursory checks rendering low performance of CP derivatives are as well as "peak values" of TF on English and Russian are suggesting little use of it, but more systematic study might get needed for final confirmation.

- The lack of available memory (32G) made it impossible to explore N>3 for smaller Chinese training corpus and N>2 for larger corpus. While smaller Chinese corpus has show N=2 providing the highest scores compared to N=1 and N=2, and larger Chinese corpus has shown N=2 to be the best, it would better be confirmed with N=3 for larger corpora, given more than 32G memory available.

- The lack of Chinese knowledge has prevented to do reliable interpretation and judgments on the F1 scores in respect to tokenization, so further exploration involving Chinese tokenization might get required to do more reliable assessment of applicability of the presented study for Chinese language.

Ethical Impact

Presented work appears to have immediate positive impact of inclusiveness in respect to the cultures relying on so-called low-resource languages and dialects which don't have enough capacity to be studied by the linguistic science. Presumably, suggested technology might simplify study of such languages, providing initial lexicon dictionaries based on raw field data and open the way for further studies of grammar of such languages relying on further extensions of this technology.

The other long-term positive ethical impact is associated with the "interpretable" nature of the approach being explored in this work, so that open, transparent, and human-friendly linguistic models can be developed for any human language and delivered to production, making sure that no automation based on "black-box" artificial intelligence might have impact on decreasing the quality of human lives.

No negative ethical impacts appear to be connected with this work.

Acknowledgments

We are grateful to Ben Goertzel and Linas Vepstas who have gave us motivation to work in the Interpretable Natural Language Processing and Unsupervised Language Learning domains, to Andres Suarez, who have helped us to collect some of the training corpora and to Nikolay Mikhaylovskiy for discussion on the results.

References

1. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. 2017. *Attention Is All You Need.* arXiv:1706.03762 [cs.CL].

2. Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei. 2020. *Language Models are Few-Shot Learners*. arXiv:2005.14165 [cs.CL].

3. Linas Vepstas, Ben Goertzel. 2014. *Learning Language from a Large (Unannotated) Corpus*. arXiv:1401.3372 [cs.CL].

4. Anton Kolonin. 2015. *Automatic text classification and property extraction*. SIBIRCON/SibMedInfo Conference Proceedings, ISBN 987-1-4673-9109-2, pp.27-31.

5. Glushchenko, A. et al. 2018. *Unsupervised Language Learning in OpenCog*. In: Iklé, M., Franz, A., Rzepka, R., Goertzel, B. (eds) Artificial General Intelligence. AGI 2018. Lecture Notes in Computer Science(), vol 10999. Springer, Cham. https://doi.org/10.1007/978-3-319-97676-1_11.

6. Glushchenko, A., Suarez, A., Kolonin, A., Goertzel, B., Baskov, O. 2019. *Programmatic Link Grammar Induction for Unsupervised Language Learning*. In: Hammer, P., Agrawal, P., Goertzel, B., Iklé, M. (eds) Artificial General Intelligence. AGI 2019. Lecture Notes in Computer Science(), vol 11654. Springer, Cham. https://doi.org/10.1007/978-3-030-27005-6_11.

7. Deniz Yuret. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. Microsoft Press, Redmond, WA. arXiv:cmp-lg/9805009.

8. Vignav Ramesh and Anton Kolonin. 2020. *Interpretable Natural Language Segmentation Based on Link Grammar*. Conference: 2020 Science and Artificial Intelligence conference (S.A.I.ence). DOI: 10.1109/S.A.I.ence50533.2020.9303220.

9. Vignav Ramesh and Anton Kolonin. 2021. *Natural Language Generation Using Link Grammar for General Conversational Intelligence*. arXiv:2105.00830 [cs.CL].

10. Vignav Ramesh and Anton Kolonin. 2022. *Unsupervised Context-Driven Question Answering Based on Link Grammar*. In: Goertzel, B., Iklé, M., Potapov, A. (eds) Artificial General Intelligence. AGI 2021. Lecture Notes in Computer Science(), vol 13154. Springer, Cham. https://doi.org/10.1007/978-3-030-93758-4_22.

11. Logan R. Kearsley. 2016. *A Hybrid Approach to Cross-Linguistic Tokenization: Morphology with Statistics*. Brigham Young University. Theses and Dissertations.

12. Jesse O. Wrenn, Peter D. Stetson, and Stephen B. Johnson. 2007. *An Unsupervised Machine Learning Approach to Segmentation of Clinician-Entered Free Text*. PubMed Central. AMIA Annu Symp Proc. 2007; 2007: 811–815. PMCID: PMC2655800 PMID: 18693949

13. Karl Friston. 2010. *The free-energy principle: a unified brain theory?*. Nat Rev Neurosci 11, 127–138 (2010). https://doi.org/10.1038/nrn2787.

14. Anton Kolonin. 2021. *Neuro-Symbolic Architecture for Experiential Learning in Discrete and Functional Environments*. In: Goertzel, B., Iklé, M., Potapov, A. (eds) Artificial General Intelligence. AGI 2021. Lecture Notes in Computer Science(), vol 13154. Springer, Cham. https://doi.org/10.1007/978-3-030-93758-4_12.

15. Anand Gopalakrishnan, Kazuki Irie, Jürgen Schmidhuber, Sjoerd van Steenkiste. 2022. *Unsupervised Learning of Temporal Abstractions with Slot-based Transformers*. arXiv:2203.13573 [cs.LG].

16. Sun, C. C., Hendrix, P., Ma, J.Q. & Baayen, R. H. (2018). *Chinese Lexical Database (CLD): A large-scale lexical database for simplified Mandarin Chinese*. Behavior Research Methods, https://doi.org/10.3758/s13428-018-1038-3.

17. Evgenii Vityaev, Anton Kolonin , Andrey Kurpatov, Artem Molchanov. 2022. *Brain Principles Programming*. arXiv:2202.12710 [q-bio.NC].

18. Junfeng Jiang, Jiahao Li. 2018. Constructing Financial Sentimental Factors in Chinese Market Using Natural Language Processing.

19. Castillo-Domenech C., Suarez-Madrigal A.: *Statistical parsing and unambiguous word representation in OpenCog's Unsupervised Language Learning project*. Göteborg : Chalmers U n i v e r s i t y o f T e c h n o l o g y (2 0 1 8) .

Appendix A. Reproducibility

Obtaining the Corpora

Chinese train corpora can be obtained as CLUE benchmark News 2016 dataset downloaded following the https://github.com/brightmart/nlp_chinese_corpus link. When downloaded, the folder called “new2016zh” will have two files called “news2016zh_valid.json” (283711020 bytes) and “news2016zh_train.json” (8930014780 bytes), corresponding to smaller and larger training datasets in scope of our work, respectively. Each of the two files have been processed by the script, parsing JSON, selecting “title”, “desc”, and “content” fields, saving each of the fields as individual line, so the two plain text files have been produced, respectively: “news2016zh_valid.txt” (269553996 bytes, 230391 lines) and “news2016zh_train.txt” (8481842006 bytes, 7292256 lines). In further discussion the corpora can be referred as “CLUE News 2016 Valid” and “CLUE News 2016 Train”, respectively.

English Brown train corpus was have been downloaded from http://www.sls.hawaii.edu/bley-vroman/brown_nolines.txt (6026059 bytes, 19810 lines). For extra test validation purposes, not presented in the paper, we have used random subsets of 100 sentences selected from the same Brown corpus. In further discussion the folder containing the downloaded files can be referred as “Brown”.

English train corpus of Gutenberg Children collection (29M size) was obtained from https://

www.gutenberg.org/, based on the books used in the Babi CBT corpus (https://research.fb.com/downloads/babi/), but fitting our purposes: just the raw text from the books downloaded manually from UTF8 links like https://www.gutenberg.org/cache/epub/35688/pg35688.txt, without their question formatting, according to [19]. In further discussion the folder containing the downloaded files can be referred as “Gutenberg Children”.

English train corpus of Gutenberg Adult collection (140M size) was obtained from https://www.gutenberg.org/, based on selection of 361 Gutenberg project (https://www.gutenberg.org/) books in (book id in range ID 53000 to 53499, with the books downloaded manually from UTF8 links like https://www.gutenberg.org/files/53000/53000-0.txt without processing. In further discussion the folder containing the downloaded files can be referred as “Gutenberg Adult”.

Russian train corpora was downloaded from https://www.kaggle.com/datasets/oldaandozerskaya/fiction-corpus-for-agebased-text-classification. The two folder called “test” (141M size) and “previews” (825M size) were used independently as alternative train corpora. In further discussion the corpora can be referred as “RusAge Test” and “RusAge Previews”, respectively.

Parallel Chinese/English corpus of 100 multisentence statements related to personal finance can be downloaded from https://magichub.com/datasets/chinese-english-parallel-corpus-finance/ to tab-delimited text file with individual columns for Chinese and English versions entitled as “zh” and “en”, respectively. The Russian extension to it, with the only one column of Russian translation entitled as “ru” can be downloaded from https://github.com/aigents/pygents/blob/main/data/corpora/Russian/magicdata/zh_en_ru_100/CORPUS_ZH_EN_RU.txt. In further discussion the file can be referred as “CORPUS_ZH_EN_RU.txt”.

Reference lexicon dictionaries for English And Russian have been downloaded as text files from https://github.com/aigents/aigents-java/blob/master/lexicon_english.txt and https://github.com/aigents/aigents-java/blob/master/lexicon_russian.txt, respectively.

Reference lexicon dictionaries for Chinese have been downloaded from links: http://www.chineselexicaldatabase.com/download.php, https://www.plecoforums.com/threads/word-frequency-list-based-on-a-15-billion-character-corpus-bcc-blcu-chinese-corpus.5859/, and http://crr.ugent.be/programs-data/subtitle-frequencies/subtlex-ch. Each of these links were used to download comma-separated or tab-separated files with lists of words representing Chinese lexicon with different attributions. The individual columns corresponding to words have been extracted along with frequencies of the words, if present, and then unified lexicon has been created for further lexicon-based parsing and lookup for accuracy of lexicon discovery.

Experimental Environment

The Pyhon 3 code used to run the experiments can be downloaded from https://github.com/aigents/pygents/tree/main/pygents, with external dependencies on Python packages: math, copy, pandas, seaborn, matplotlib, html, urllib, abc, pickle, re, jieba, and numpy. In order to run the following code, four imports are expected, as follows:

```
from pygents.token import *
from pygents.text import *
from pygents.util import *
from pygents.plot import *
```

The experiments have been run using Python 3.7.7 virtual environment on MacBook Pro with 2,9 GHz Intel Core i9 Processor, 32G RAM, and macOS High Sierra.

Model Building

To run the model building on Chinese CLUE News 2016 Valid and CLUE News 2016 Train corpora for Chinese, the following code has been used to perform line-by line training on single file, using the FreedomTokenizer class from token.py module. The counted numbers of parameters, corresponding to number of weights or frequency count for either N-gram or transition between N-grams, have been found to be 143129564 (corresponding to Valid corpus and N={1,2,3}) and 249859247 (corresponding to Train corpus and N={1,2})

```
max_n = 3 # 2 for larger Train corpus, 3 for smaller Valid corpus
zh_chars=FreedomTokenizer(max_n=max_n,mode='chars',debug=False)
    with open(join(path, <corpus file name>),errors='ignore') as f:
        while True:
            line = f.readline()
            if not line:
                break
            zh_chars.train([line])
zh_chars.store(<model file name>)
print(zh_chars.count_params())
```

To run the model building on English Brown corpus the following code has been used to train the model on entire corpus at once, the number of parameters have been found to be 52502749.

```
brown_text = url_text("http://www.sls.hawaii.edu/bley-vroman/brown_nolines.txt")
brown_chars=FreedomTokenizer(<model file name>,max_n=7,mode='chars',debug=False)
brown_chars.train([brown_text])
brown_chars.store(<model file name>)
print(brown_chars.count_params())
```

To run the model building on English Gutenberg Children and Adult corpora the following code has been used to train the model on entire corpus at once, the number of model parameters have been found to be 12321620 and 44900866, respectively.

```
def tokenizer_train_folder(t,path):
    onlyfiles = [f for f in listdir(path) if isfile(join(path, f))]
    for file in onlyfiles:
        with open(join(path, file),errors='ignore') as f:
            lines = f.readlines()
            t.train(lines)
mode = 'chars'
child_chars = FreedomTokenizer(max_n=7,mode=mode,debug=False)
tokenizer_train_folder(child_chars,<coprus folder name>)
child_chars.store(<model file name>)
print(child_chars.count_params())
```

Model building is done my the *train* method of *FreedomTokenizer* class in *token.py* module which is calling *grams_count_with_gram_freedoms* residing in *text.py* module internally.

Two different kind of models could be built based on *mode* parameter which could be either *'chars'* or *'grams*', according to either N-gram-to-N-char or N-gram-to-N-gram option, respectively.

The same code above was used to run the model building on RusAge Test and Previews corpora, the number of model parameters have been found to be 28998065 and 207808799, respectively.

Each model building was taking up to one hour for smaller corpora and up to several hours for larger models. While building the CLUE News 2016 Valid model the maximum N-gram rank was N=3 and for CLUE News 2016 Train in was N=2, due to the available memory limitations. Building the largest model based on CLUE News 2016 Train with maximum N-gram rank N=2 took 11 hours which was the maximum training time.

Performing Tokenization

All tokenization experiments have been run with help of function called *evaluate_freedom_tokenizer_options* (*token_plot.py* module) passing the tokenization class called *FreedomBasedTokenizer (token.py module)* supplied with metrics used for tokenization in forward and backward directions, list of different combinations of N of ngram ranks and list of tokenization thresholds, like in the following example used to perform Tokenization for English.

```
test_df=pd.read_csv(os.path.join(path,'CORPUS_ZH_EN_RU.txt'),delimiter='\t')
test_texts = list(test_df['en']) # column could be 'en', 'zh' or 'ru'
ref_tokenizer = DelimiterTokenizer()
ngram_params = [[1],[2],[3],[4],[5],[6],[7],[1,2],[2,3],[1,2,3],[1,2,3,4],[4,5,6,7],[1,2,3,4,5],
[1,2,3,4,5,6,7]]
compression_thresholds = [0,0.0001,0.001,0.01,0.1]
tokenization_thresholds = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
base=FreedomTokenizer(name=<model file name>,max_n=7,mode='chars',debug=False)
title = 'F1 - Brown ddf- & ddf+'
for filter_threshold in compression_thresholds:
    if filter_threshold > 0:
        model_compress_with_loss(base.model,filter_threshold)
    parameters = base.count_params()
    title="{} filter={} parameters={}".format(title,filter_threshold,parameters)
    evaluate_freedom_tokenizer_options(test_texts,ref_tokenizer,FreedomBasedTokenizer(ba
se,'ddf-','ddf+'),ngram_params,tokenization_thresholds,title=title)
```

Hyper-parameters for metrics passed to the *FreedomBasedTokenizer* class constructor above could be 'p-','p+' for conditional probabilities (CP) in forward and backward directions, 'dp-','dp+' for derivatives of CP, 'dvp-','dvp+' for variances of CP, 'f-','f+' for transition freedoms (TP) in forward and backward directions, 'df-','df+' for derivatives of TP, 'dvf-','dvf+' for variances of TP, and 'peak-','peak+' for variances of TP.

For English and Russian, the reference *DelimiterTokeinzer* (*tokep.py* module) was used for tokenization being rule-based, separating words by spaces and detaching any punctuation marks, counting the latter along with spaces and words as individual tokens.

```
ref_tokenizer = DelimiterTokenizer()
```

The following combinations of N-gram ranks, model compression thresholds and tokenization thresholds have been used as hyper-parameters for English and Russian.

```
ngram_params = [[1],[2],[3],[4],[5],[6],[7],[1,2],[2,3],[1,2,3],[1,2,3,4],[4,5,6,7],[1,2,3,4,5],
[1,2,3,4,5,6,7]]
compression_thresholds = [0,0.0001,0.001,0.01,0.1]
tokenization_thresholds = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
```

For Chinese, reference *JiebaTokenizer* (*token.py* module) was used.

```
ref_tokenizer = JiebaTokenizer()
```

The following combinations of N-gram ranks, model compression thresholds and tokenization thresholds have been used as hyper-parameters for Chinese.

```
ngram_params = [[1],[2],[3],[1,2],[2,3],[1,2,3]]
compression_thresholds = [0,0.0001,0.001,0.01,0.1]
tokenization_thresholds = [0.0001,0.0005,0.001,0.005,0.01,0.02,0.05,0.1,0.2,0.4,0.8]
```

All sets of hyper-parameters, including metrics based on CP and TF, different N-gram ranks, model compression thresholds and tokenization thresholds have been applied for different models across all languages against the same CORPUS_ZH_EN_RU.txt test set.

Additionally, for validation purposes, in order to confirm reliability of hyper-parameters providing the best F1 scores, the same tokenization experiments have been run using different splits of the CORPUS_ZH_EN_RU.txt (all 100 lines, first 50 lines, last 50 lines) as well as random 100 lines selected from Brown corpus ensuring that the same hyper-parameters were providing the highest F1 scores with close score values.

Every tokenization trial task for individual pre-built model, given the selected tokenization metrics, for 3-dimensional grid of hyper-parameter search, including *ngram_params*, *compression_thresholds* and *tokenization_thresholds* was taking no more than two hours per trial with one hour as an average.

Evaluation

The *evaluate_freedom_tokenizer_options* function used to run the experiments above was performing F1 score assessment internally using *evaluate_tokenizer_f1* function (*token.py* module) which was accessing average F1 across all input test texts comparing outputs of the evaluated and reference tokenizers.

Evaluation of lexicon-based tokenization for reference have been done with the following code, merging reference lexicon dictionary with list of conventional punctuation symbols and using *LexiconIndexedTokenizer* class (*token.py* module). The sorting modes of it are corresponding to do greedy search based on either token length (0), token frequency (1) or multiplication or token length and logarithm of frequency (2).

```
test_df=pd.read_csv(os.path.join(path,'CORPUS_ZH_EN_RU.txt'),delimiter='\t')
test_texts = list(test_df['en']) # column could be 'en', 'zh' or 'ru'
ref_tokenizer = DelimiterTokenizer() # use JiebaTokenizer() for Chinese
#get raw lexicon list, use respective source lexicons for English/Russian/Chinese
en_lex = list(pd.read_csv("https://raw.githubusercontent.com/aigents/aigents-java/master/lexicon_english.txt",sep='\t',header=None,na_filter=False).to_records(index=False))
# add delimiters to the list
delimiters = ' \t\n\r\'`"“”+=-_&/|\*()[]<>#^@~,;:.!?'
lex = en_lex + [(i, top_weight) for i in list(delimiters)]
en_lex0_tokenizer = LexiconIndexedTokenizer(lexicon=lex,sortmode=0,cased=True)
en_lex1_tokenizer = LexiconIndexedTokenizer(lexicon=lex,sortmode=1,cased=True)
en_lex2_tokenizer = LexiconIndexedTokenizer(lexicon=lex,sortmode=2,cased=True)
print(t,en_lex0_tokenizer.count_params())
print(evaluate_tokenizer_f1(test_texts,del_tokenizer,en_lex0_tokenizer,debug=False))#sort by token length
print(evaluate_tokenizer_f1(test_texts,del_tokenizer,en_lex1_tokenizer,debug=False))#sort by frequency
print(evaluate_tokenizer_f1(test_texts,del_tokenizer,en_lex2_tokenizer,debug=False))#sort by token length and frequency
```

Evaluation of lexicon discovery precision have been achieved with extra parameters passed to the *evaluate_tokenizer_f1* function (*token.py module*), so that all tokens identified by evaluated (freedom-based) and reference (lexicon-based or rule-based) tokenizers could be collected, as follows. Upon the collection of the actual (evaluated tokenizer) and expected (reference tokenizer), the precisions of both actual and expected counts of tokens have been computed referring to relevant tokens actually existing in the reference lexicon.

```
base=FreedomTokenizer(name=<model file name>,max_n=7,mode='chars',debug=False)
model_compress_with_loss(base.model,0.0001)
test_tokenizer = FreedomBasedTokenizer(base,'dvf-','dvf+') # for English and Russian
test_tokenizer.set_options(nlist = [1], threshold=0.4)  # for English and Russian

expected = {}
actual = {}
tokenization_f1 = evaluate_tokenizer_f1(test_texts,del_tokenizer,test_tokenizer,expected_collector=expected,actual_collector=actual)

expected_count = sum([expected[key] for key in expected])
relevant_count = sum([expected[key] for key in expected if key.lower() in en_lex_delimited_dict])
irrelevant_count = sum([expected[key] for key in expected if not key.lower() in en_lex_delimited_dict])
print(expected_count,relevant_count,irrelevant_count,relevant_count/expected_count,(relevant_count)/expected_count)

actual_count = sum([actual[key] for key in actual])
relevant_count = sum([actual[key] for key in actual if key.lower() in en_lex_delimited_dict])
irrelevant_count = sum([actual[key] for key in actual if not key.lower() in en_lex_delimited_dict])
print(actual_count,relevant_count,irrelevant_count,relevant_count/actual_count,(relevant_count)/actual_count)
```

Reproducibility Summary

The following addresses the Dodge et al, 2019 and Joelle Pineau's reproducibility checklist.

A clear description of the mathematical setting, algorithm, and/or model: the metrics and models described in [11,12] were used and extended as described in the paper.

Source code, with specification of all dependencies, including external libraries: see Appendix A below with public code references.

Description of computing infrastructure used: MacBook Pro 2018, Processor 2,9 GHz Intel Core i9, 32 GB 2400 MHz DDR4, Macintosh HD 2TB.

The average runtime for each model or algorithm (e.g., training, inference, etc.), or estimated energy cost: Model building per corpus – 1-11 hours, corresponding 300-3300 watts. Tokenization per single hyper-parameter search trial for one specific model, the test sent, single tokenization metric and 3D grid of 3 parameters was

taking less than 2 hours, corresponding to 600 watts of energy consumption.

Number of parameters in each model: Chinese 143M and 250M, English 12M and 45M, Russian 29M and 208M – all for smaller and larger corpora, respectively.

Corresponding validation performance for each reported test result: The cross validation has been performed in tow completely different yet complementary ways. First, we have used different models built upon different independent data sets (smaller and larger corpora) against the same test set independent from the models. In this kind of validation scenario, the best tokenization metrics and N-gram ranks for the best-performing configurations were the same across different corpora and splits for specific language; model compression threshold in range 0 to 0.01 provided less than 2% variance or the best performing-configurations across corpora and splits for specific language; tokenization threshold best performing for a language provided less than 3% variance. Second, as mentioned in section 3.5 of the main paper body, we have run the tokenization on the same model with original test set of 100 sentence split in two independent sub-sets of 50 sentences each – in this kind of validation scenario the difference of F1 score for the best-performing configurations of hyper-parameters has turned to be less than 1% (for English the F1 has happened to be 0.99 and 0.99 for both test sub-sets).

Explanation of evaluation metrics used, with links to code: F1 score assessment internally using *evaluate_tokenizer_f1* function (*token.py* module) relying on set-based F1 assessment in *calc_f1* function (*util.py* module).

The exact number of training and evaluation runs: given the deterministic nature of model building and tokenization algorithms, so that the same run with the same training corpus, model built and the same tokenization hyper-parameters were providing reproducible results, only the difference in training set or a test model was considered so each training set (smaller and larger) were given single "clean" run with certain number of trial/debugging runs before the final one.

Bounds for each hyper-parameter: were selected in ranges: N rank – 1 to 7, model compression threshold – 0 to 0.1, tokenization threshold – 0 to 0.9.

Hyper-parameter configurations for best-performing models: reported in section 4.1, 4.2 and 4.3 in the main body of the paper.

Number of hyper-parameter search trials: the goal of the presented study was to find the top F1 score metrics possible for completely unsupervised tokenization along with the best-performing hype-parameters, so the same unique combination of training corpus, test corpus (or sub-split of it) and combination of hyper-parameters were given exactly one final "clean" run (not counting certain number of trial/debugging runs before the final one).

The method of choosing hyper-parameter values (e.g., uniform sampling, manual tuning, etc.) and the criterion used to select among them (e.g., accuracy): used 3D grid search for hyper-parameters with the grid parameters adjusted according to experimental trial/debugging runs to cover the hyper-parameter space where the target F1 is subject to change.

Summary statistics of the results (e.g., mean, variance, error bars, etc.): given the deterministic nature of model building and reproducibility of tokenization algorithm, only the difference in training set (smaller and larger corpus) or a test set (full 100 sentence or either of 50/50 splits) was considered, the variance of F1 score within intervals of best-performing hyper-parameters across corpora and splits was under 3%.

Relevant details such as languages, and number of examples: 3 languages each with two train corpora and 1 independent test corpora with two 50/50 splits as described in the body of the main paper and above.

Details of train/validation/test splits: described in the body of the main paper and above.

Explanation of any data that were excluded, and all pre-processing steps: no data was excluded:

Data or link to a downloadable version of the data: provided in the above of the appendix.

For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control: no newly collected data have been discussed in the scope of this paper.

Appendix A. Public Code References

Repository
https://github.com/aigents/pygents

Slides:
https://github.com/aigents/pygents/blob/main/docs/2022/experiential-sequential-2022.pdf

Basic tokenization exploration notebooks:
https://github.com/aigents/pygents/tree/main/notebooks/nlp

English tokenization exploration notebooks
https://github.com/aigents/pygents/tree/main/notebooks/nlp/english

Russian tokenization exploration notebooks
https://github.com/aigents/pygents/tree/main/notebooks/nlp/russian

Chinese tokenization exploration notebooks
https://github.com/aigents/pygents/tree/main/notebooks/nlp/chinese