Coreference Resolution Algorithm for Ukrainian-Language Texts Using Decision Trees

Sergiy Pogorilyy [1], Pavlo Biletskyi [1]

[1] *Taras Shevchenko National University of Kyiv, 60 Volodymyrska Street, Kyiv, 01033, Ukraine*

Abstract

The paper examines the problem of coreference resolution in Ukrainian-language texts using decision trees. An application that uses vector representations of Elmo words and other characteristics for the automated formation of a decision tree has been developed. A set of prepared texts containing more than 360,000 words was used to form the decision tree and evaluate the accuracy of the algorithm. The decision tree created to determine whether a pair of objects is coreference was used to form clusters of coreference objects. Special metrics were used for comparison with the results obtained by other algorithms in the Ukrainian language.

Keywords

Coreference resolution, natural language processing (NLP), decision trees, artificial intelligence (AI), vector words representation, neural networks.

1. Introduction

Natural language processing (NLP) is a large field that includes many tasks: translation between natural languages, human-computer interfaces, analysis and generation of natural speech, information extraction. One of the tasks of natural language processing is coreference resolution. Coreferentiality in texts means a relationship between syntactic units indicating the same object (referent) in a given context [1].

Below are examples of coreference (the referent is highlighted in bold, the pronouns are underlined).

Simple anaphora example:

*He crossed the **mountain**. It was high.*

Simple cataphora example:

*She took the road to the right. **Mery** was in a good mood today.*

An example of a compound antecedent:

***John, David and Julia** were tired. They all worked underground.*

In comparison with other European languages - English, German, French, Italian, Spanish - the Ukrainian language has an arbitrary word order, just like other Slavic languages - Polish, Russian, Serbian, Croatian, Romanian. For example, for the Ukrainian language:

*Karpo prykynuv take slivtse, shcho **batko** perestav struhaty i pochav pryslukhatys. Vin hlianuv na syniv cherez khvorostianu stinu. Syny stoialy bez dila y balakaly, pospyravshys na zastupy (Ivan Nechui-Levytskyi, «Kaidasheva simia», original word order).*

From the second sentence in the example, it is possible to rearrange the words to form other grammatically correct sentences with very close meanings, but with different accents, depending on the author's intention:

- Cherez khvorostianu stinu vin hlianuv na syniv.

13th International Scientific and Practical Conference from Programming UkrPROGP'2022, October 11-12, 2022, Kyiv, Ukraine
EMAIL: sdp77@i.ua (A. 1); 1234bpv@i.ua (A. 2)
ORCID: 0000-0002-6497-5056 (A. 1); 0000-0001-5425-3706 (A. 2)

- Na syniv vin hlianuv cherez khvorostianu stinu.
- Cherez stinu khvorostianu vin hlianuv na syniv.
- Cherez khvorostianu stinu hlianuv vin na syniv.
- Cherez khvorostianu stinu na syniv hlianuv vin.

At the same time, only one combination is possible in English: *«He looked at his sons through the twig wall»*, because it uses the standard subject-verb-object (SVO, subject verb object) word order. In other languages, the subject-object-verb (SOV, subject object verb) order is also common. In this regard, the algorithm for the Ukrainian language should be able to work with different word orders.

Coreference resolution allows finding connections between sentences and within them, extracting information from texts, improving the results of text analysis in other tasks, such as translation from one language to another, assessment of the coherence of texts.

At the initial stages of research, algorithms based on rules manually formed by experienced linguists were used to search for coreference objects. Such algorithms were created for a specific language and had to take into account many features to achieve high results.

Over time, automated approaches such as neural networks and decision trees began to be used to solve the problem. They do not require manual rule creation, but require large data sets to prepare them. Nevertheless, often, automated algorithms use simple rules to form initial clusters.

In previous works, such as [2], [3], usage of the decision trees was considered for coreference resolution and utilized on MUC-6 dataset [2] for English language and CoNLL-2012 [3] for Arabic, Chinese and English, but decision trees appliance was newer tested on Ukrainian-language datasets.

The article proposes a method for coreference resolution in Ukrainian language using decision trees. An application has been developed that uses Elmo vector representations of words and other features for automated decision tree formation. A set of prepared texts containing more than 360,000 words was used to form the tree and evaluate the accuracy of the algorithm. The tree formation parameters are selected. The decision tree created to determine whether a pair of objects is coreference used to form clusters of coreference objects. Special metrics were used for comparison with the results obtained by other algorithms in the Ukrainian language.

2. Means used to implement the coreference resolution algorithm

The application uses vector representations of words obtained by applying the ELMo library[4]. This library allows converting words into vectors corresponding to their semantic, lexical, and syntactic meaning. Unlike other libraries that allow to form vector representations of words, such as Word2Vec[5], vector representations formed by ELMo take into account not only the meaning of a single word, but also the meaning of surrounding words, which allows to better find connections between individual words and sentences. ELMo uses neural networks to obtain vector representations of words and requires training. Version used in application adapted for the Ukrainian language.

The Scikit-learn library [6] is used. This library includes many tools for solving regression, clustering and classification problems, in particular, it contains an optimized implementation of decision trees with the ability to configure tree construction parameters.

One of the main advantages of decision trees over other algorithms is the ability to visualize the constructed tree. This allows conducting analysis of created tree and it`s internal operational logics. Also, decisions automatically made by the tree on each of its steps allow to improve the analysis of data, find dependencies between parameters, determine the limit of overfitting and adjust parameters for forming decision tree. Furthermore, after analysis completed, it is possible to make changes in decision tree structure and see, if changes are improving the result. For this, the Graphviz library [7], which is specifically developed for graph visualization was used.

A prepared corpus of Ukrainian-language texts containing more than 360,000 words (> 2,500 texts) was used to create a decision tree. The marking of coreference objects in it is carried out manually, and for obtaining additional information (gender, number, lemmatized (initial) version of the word) the UDpipe library [8], which uses neural networks and is trained on the Ukrainian text corpus is used.

3. Data format for representing coreference objects and their analysis

To analyze texts using decision trees, they need to be presented in the correct form. The following characteristics are used in the work to describe coreference objects (considered in the article [8]):

- Cosine similarity of vectors of the semantic representation of objects under consideration. For this, a pre-trained ELMo model was used. In the work, if the number of words included in the object > 1, the arithmetic mean of word vectors is used.
- The number of words between the selected objects.
- The number of objects between the selected potentially coreference objects.
- Boolean, true if the first object is a pronoun.
- Boolean, true if the second object is a pronoun.
- Boolean, true if the lemmatized versions (initial word forms) of the objects match. In the algorithm, the matching of lemmatized versions is defined as the matching of at least one word in both objects under consideration.
- Boolean, true if both objects have the same number (singular or plural).
- Boolean, true if both objects have the same genus.
- Boolean, true if both objects are proper names.

The input of the algorithm is a text consisting of individual words, punctuation, and additional information prepared using the UDpipe library. This includes gender, number, lemmatized version of the word, part of speech. Also, for words included in coreference groups, an identifier is added, which allows to assign a specific word or word combination to a coreference group (prepared manually).

For the algorithm operation, a Python list is formed for each text under consideration, containing the indexes of the words included in the word combinations, which are potentially coreferential objects, before combining them into coreferential clusters, but in a format that facilitates their subsequent union (1), w – is a separate word, $[w_{i,1}, w_{i,2} \dots]$ - is a word combination.

$$ObjectsList = [\dots [[w_{i,1}, w_{i,2} \dots]], [[w_{i+1,1}, w_{i+1,2} \dots]] \dots] \quad (1)$$

Also, lists containing correctly formed clusters of coreference objects are created. These lists are necessary at the stage of comparing the clusters obtained as a result of predictions of the decision tree and valid clusters (2), C_1, $[[w_{i,1}, w_{i,2}], [w_{j,1}, w_{j,2}]]$ - is a separate cluster.

$$CorrectObjectsList = [C_1, C_2, \dots [[w_{i,1}, w_{i,2} \dots], [w_{j,1}, w_{j,2} \dots] \dots], [[w_{i+1,1}, w_{i+1,2} \dots] \dots] \dots] \quad (2)$$

Since the clustering task is reduced in the algorithm to a classification task, to create decision trees and predictions with their help, lists with the parameters of each pair of potentially coreferential objects (3) are needed, which include cosSim – the cosine similarity of objects, nWBtw – the number of words between objects under consideration, nObjBetw – the number of objects between the objects under consideration, len1 – the length (number of words) of the first object, len2 – the length of the second object, 1pron – whether the first object is a pronoun, 2pron – whether the second object is a pronoun, 1prp – whether the first object is a proper name, 2prp – whether the second object is a proper name, lemS – whether the lemmatized versions of the objects match, gendS – whether the objects have the same gender , numS– whether the objects have the same number.

$$x = [cosSim_1, nWBtw_1, nObjBtw_1, len1_1, len2_1, 1prn_1, 2prn_1, 1prp_1, 2prp_1, lemS_1, gndS_1, nS_1] \quad (3)$$

Each pair of objects is denoted as x (4).

$$X = [x_1, x_2, x, \dots] \quad (4)$$

Labels indicating whether the pair of objects under consideration are coreferential are also required to check algorithm (5).

$$Y = [y_1, y_2, y_3, \dots] \quad (5)$$

After creating a decision tree, when using it for predictions from list (1), a list (5) containing a list of predicted clusters is generated.

$$PredictedObjectsList = [C_1, C_2, \dots [[w_{i,1}, w_{i,2} \dots], [w_{j,1}, w_{j,2} \dots] \dots], [[w_{i+1,1}, w_{i+1,2} \dots] \dots] \dots] \quad (6)$$

The resulting lists with characteristics of groups of coreference objects (3) as well as their labeling (4), containing 2,400,000 samples of coreference and non-coreference objects, are divided into two parts - the first (1500 texts, ~ 60%) is used to form a decision tree, the second (1015 texts, ~ 40%) – to check the effectiveness of the algorithm (analysis of the obtained results).

4. Algorithm for coreference resolution using decision trees

4.1. Formation of the decision tree

The decision tree is implemented using the sklearn library, which has the decision tree class sklearn.tree.DecisionTreeClassifier. When creating an instance of a class, the parameters of the decision tree are determined - the criteria for splitting into subtrees during formation, the maximum depth of the tree, the minimum number of elements for splitting, the minimum number of elements in one leaf of the tree, weights for the expected classes, and others. By default, the parameters of the tree are specified to obtain an error-free configuration of the tree on the sample used for its formation. This approach leads to excessive adaptation of the tree to the data used in its formation and reduces the accuracy of work on sets that were not previously analyzed by the algorithm. Also, for large volumes of data, the use of such a configuration creates an excessively large tree, the formation of which takes a long time.

Thus, it becomes necessary to limit the size of the tree in order to achieve higher results on the data that were not used to form the tree. For this, the parameter min_impurity_decrease was used, which allows determining the minimally sufficient value of reducing heterogeneity in subsequent subtrees during splitting. Unlike other ways of limiting the size of the tree, such as limiting the depth or limiting the number of elements in the leaves, this indicator allows more evenly limit the size of the tree. For the formation of final version of the decision tree, parameter of min_impurity_decrease was set to 0.000003.

To form a tree, prepared data of pairs of potentially coreferential objects (3) with a marking of whether they are coreferential (4) are submitted to the fit function. The output is a tree capable of analyzing pairs of coreference objects. Thus, the problem of coreference object clustering is reduced to the problem of decision tree classification.

Part of the resulting tree is shown in Fig. 1. For illustration purposes, the depth of the tree is artificially limited to fit the resulting tree on the screen. As can be seen from the figure, the division into coreferential and non-coreferential objects begins with the characteristic that allows the best separation of the current group of objects - the coincidence of lemmatized versions of objects. Further, thanks to the use of decision trees, the following logic can be followed in the subtree of the tree (Fig. 2) – if the lemmatized versions of the objects match, the length of the first and second objects is 1, and the first object is a proper name, then with a high probability a pair of objects is coreferential.

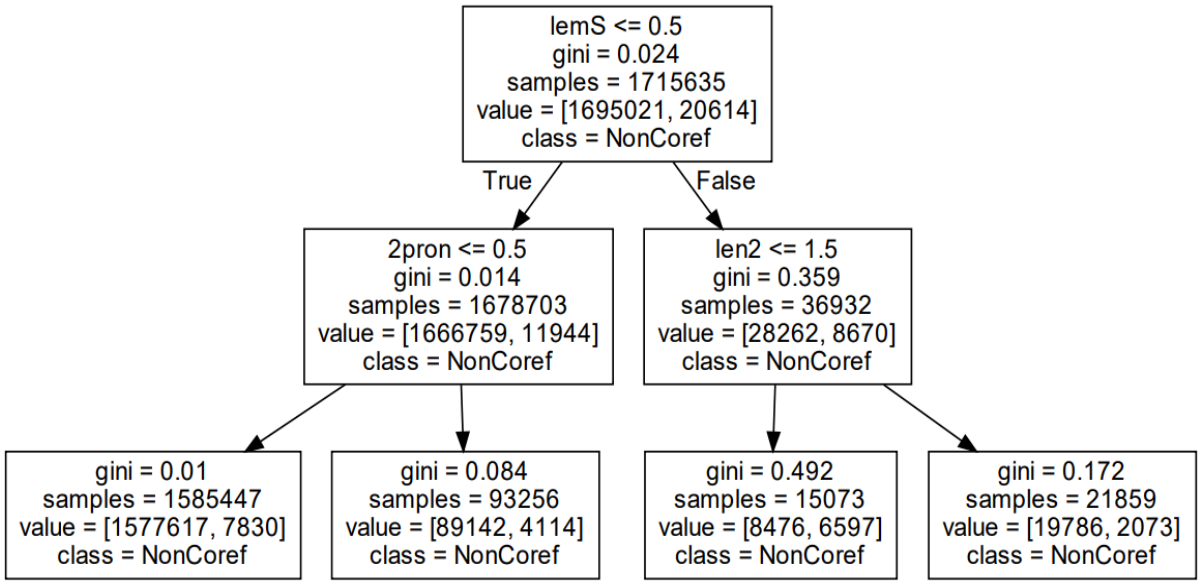

Figure 1: Subtree of the decision tree (1)

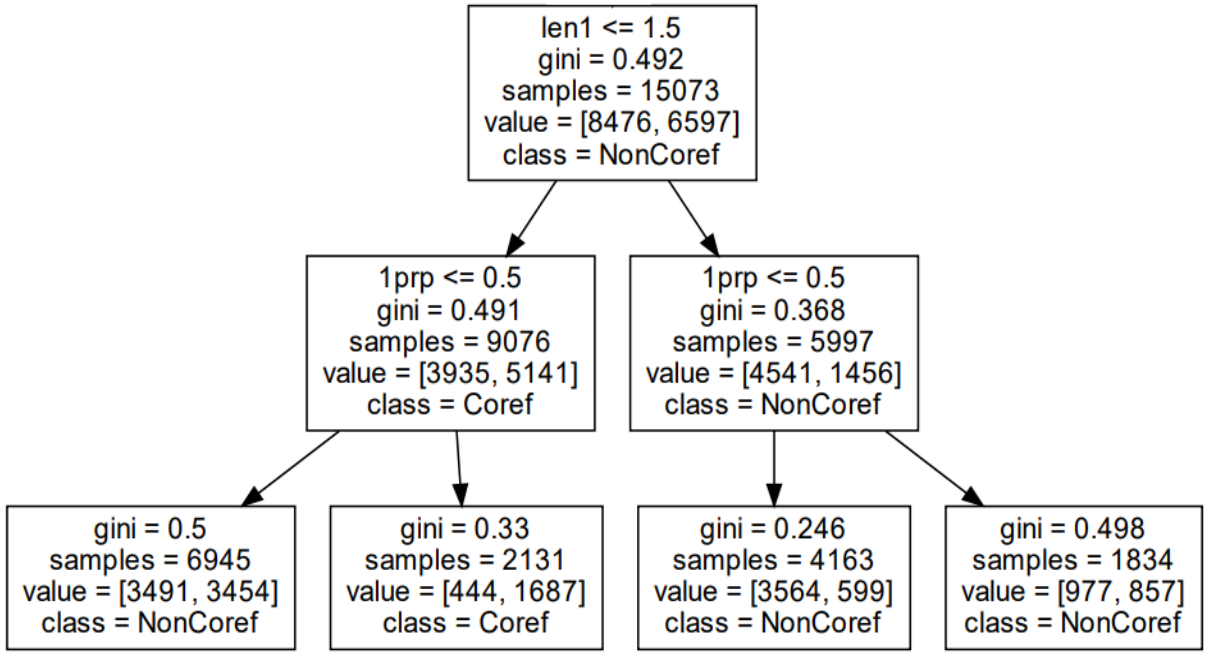

Figure 2: Subtree of the decision tree (2)

Further filtering allows finding more coreferent objects: in Fig. 3 if first object is not proper noun but pronoun, then it is considered coreferent to the second object. On Fig. 4 if first object is proper noun and its length is 2, then it is also considered coreferent.

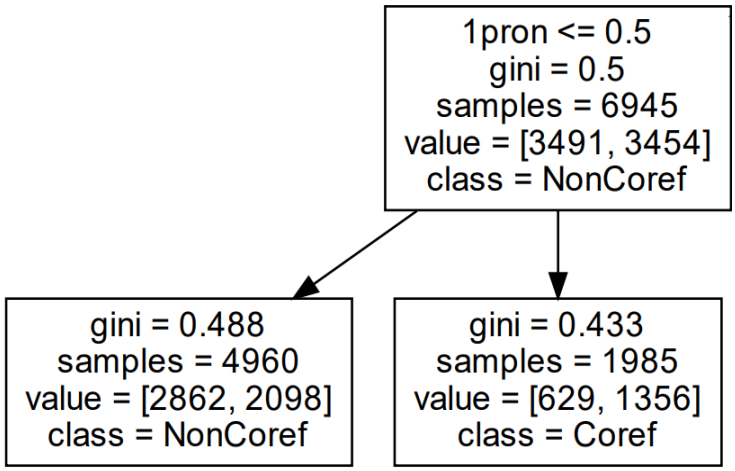

Figure 3: Subtree of the decision tree (3)

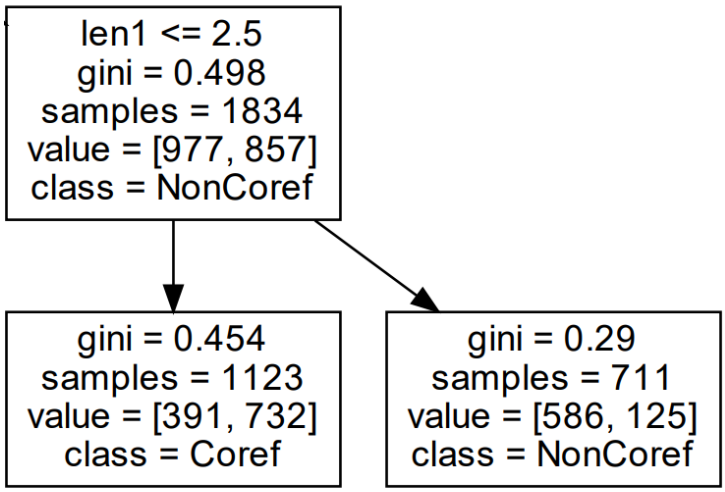

Figure 4: Subtree of the decision tree (4)

Decision trees are also allow filtering of negative coreference examples, as it is shown at Fig. 5, on which there is subtree of the decision tree right after initial filtering if lemmatized versions of objects are not same. If second object is not pronoun and cosine similarity measure of objects is lower then 0.426, then with high probability objects are not coreferent.

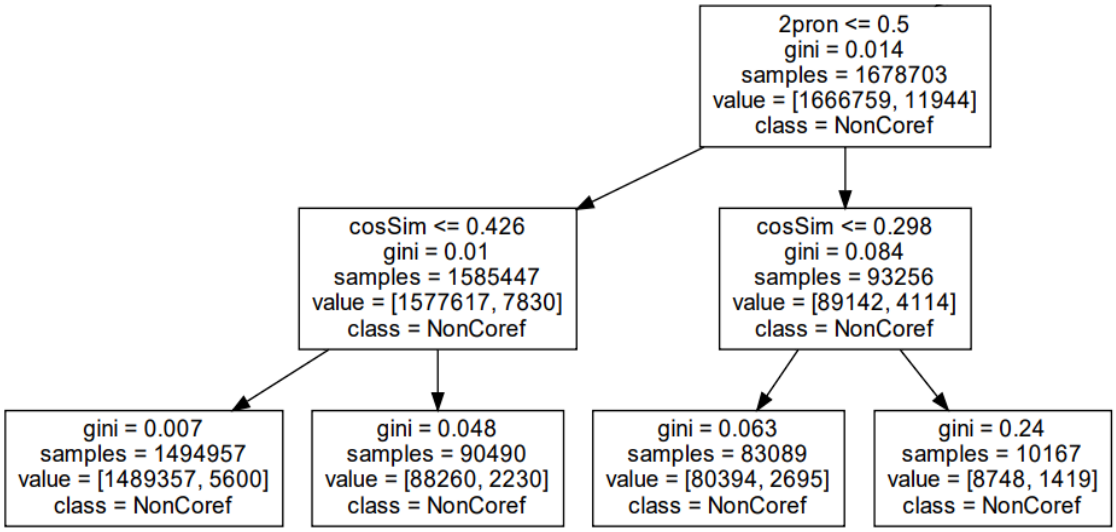

Figure 5: Subtree of the decision tree (5)

Entire decision tree is too large to show at figure, that is why to limit its size, parameter min_impurity_decrease was set to 0.00005. Resulting tree is shown at the Fig. 6. This variant of the tree shows similar performance to the tree presented in results on b^3 metric, but for the MUC metric results are lower for recall – 19.24, which also decreased F1 score to 30.77 percent (in comparison with results from table 1).

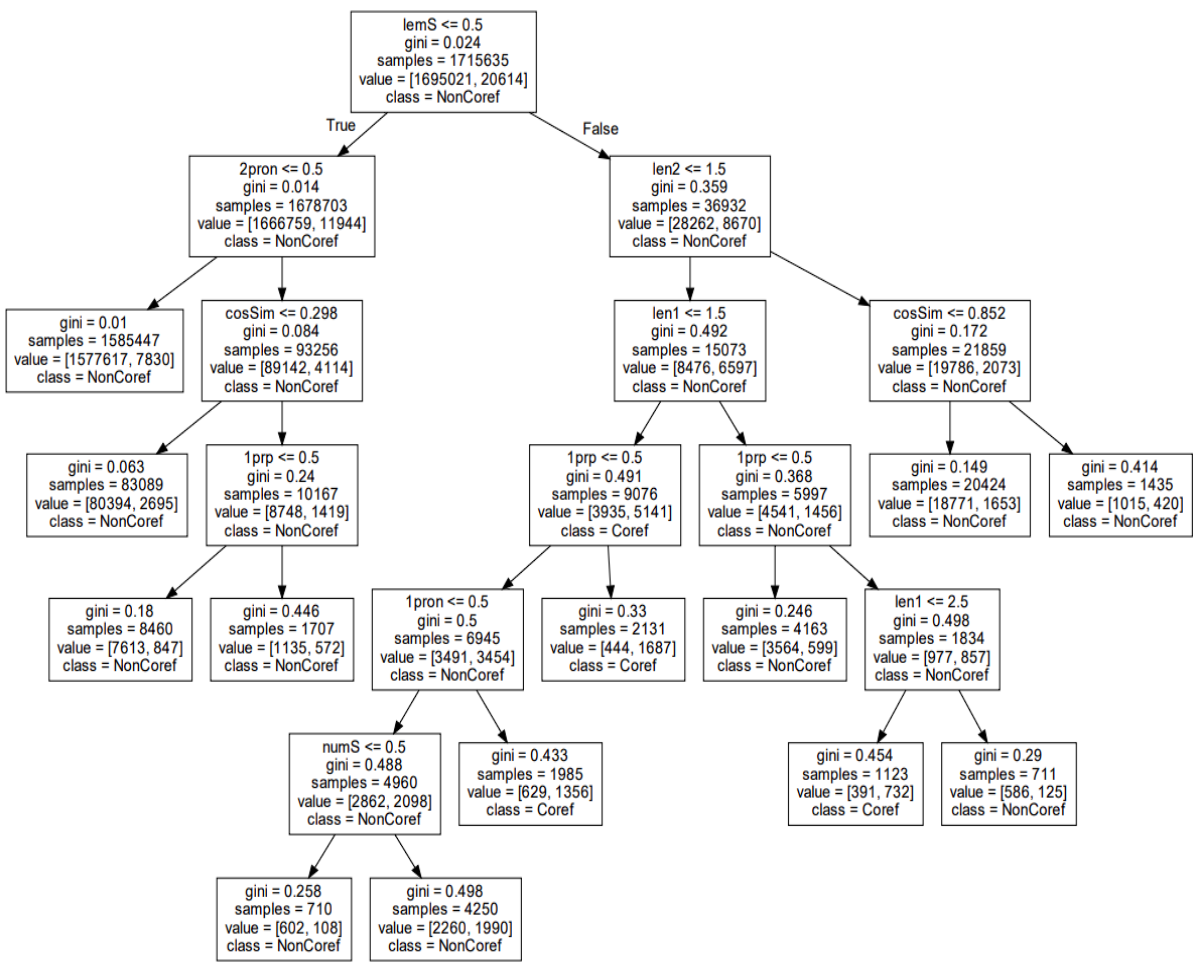

Figure 6: Decision tree

4.2. Using a decision tree for coreference resolution

After forming a decision tree, the created tree used to obtain clusters of coreference objects: pairs of objects - candidates for coreference are considered (1). For each pair of objects, the parameters (3) necessary for the classification of objects by a decision tree are determined. If the pair is classified to be coreferential, the objects are merged. After the formation of clusters containing several objects, their merging occurs if at least one pair of objects from the first and second clusters is recognized to be coreferential. During the experimental studies, it was found that when several cycles of passes are used, while cluster merging is possible, the results on the metrics (next paragraph) were 2-5% higher than without the use of cyclic passes, therefore, in the final version of the algorithm, several passes are used, while during cycle there is at least one merging. As a result of the algorithm, a list of clusters containing coreference objects within common clusters is formed (5).

5. Analysis of the obtained results

The metrics b^3 [9] and MUC [10] are used to evaluate the results obtained while using the coreference resolution algorithm. These metrics allow comparing groups of clusters - with the correct ordering of coreference objects (2) and with the predicted ordering (5), numerically reflecting the difference between clusters. For each metric, the precision (ratio of correctly selected objects to all selected objects), recall (ratio of correctly selected objects to all objects belonging to this cluster), and F1 measure (mean harmonic of precision and recall) shown.

The b^3 metric is used in a wide range of clustering problems. The b^3 metric considers individual elements in the list of predicted clusters, for which the integral value is calculated. The precision for the b^3 metric is defined as the arithmetic average of the accuracy for each element:

$$P = \frac{1}{N}\sum_N \frac{Z_n}{T_n} \tag{7}$$

where n - number of the selected element, N – number of elements in the list, Z_n - number of elements that belong to the same coreference group as the selected element (including the selected element) and are included in the predicted cluster, T_n - total number of elements in the predicted cluster.

Recall for b^3 metric is defined as the arithmetic mean of recall for each element:

$$R = \frac{1}{N}\sum_N \frac{Z_n}{M_n} \tag{8}$$

where n - number of the selected element, N - number of elements in the list, Z_n - number of elements that belong to the same coreference group as the selected element (including the selected element) and are included in the expected cluster, M_n - total number of elements in the same group as the selected element.

The MUC metric is specially designed to evaluate the performance of algorithms that solve coreference resolution task. The MUC metric considers the entire list of clusters for which indications are calculated. For MUC, recall is determined by the formula:

$$R = \frac{\sum(|S_i| - |p(S_i)|)}{\sum|S_i|} \tag{9}$$

where $|S_i|$ – number of elements in the true coreference cluster, $|p(S_i)|$ - number of subgroups into which the true coreference cluster is divided by the assumed clusters. Precision is defined as:

$$P = \frac{\Sigma(|S_i'| - |p'(S_i')|)}{\Sigma|S_i'|} \quad (10)$$

where $|S_i'|$ – the number of elements in the assumed coreference cluster, $|p'(S_i')|$ - the number of subgroups into which the true clusters divide by the assumed coreference cluster.

For each metric, the F1 measure is calculated, determined by the formula:

$$F_1 = \frac{2*R*P}{R+P} \quad (11)$$

where R - recall, P - precision.

The evaluation of results of the obtained decision tree is performed on the part of the corpus that was not used during trees formation (1015 texts). During experimental research, the optimal value of the min_impurity_decrease parameter was determined, for which the highest results on the b^3 and MUC metrics were achieved. The results of a comparison of decision trees algorithm with other algorithms used for the analysis of Ukrainian-language texts [1, 9] are shown in the table. 1.

For the metric b^3, the decision tree shows highest results compared to other approaches. On the MUC metric, the decision tree approach shows significantly higher results than those achieved using the convolutional neural network (CNN) [9] and the natural language model RoBERTa [1], similar to a two-way neural network with long and short-term memory (BiLSTM) [9]. It is worth noting that the algorithm's precision for this decision tree on the MUC and b^3 metrics is the highest compared to other approaches, while the completeness similar to the BiLSTM variant with one pass. Parameters of the decision tree formation allow to increase its precision by reducing the recall or vice versa, thus adapting to cases, for which accuracy for found coreference objects (precision) or finding the majority of coreference objects (recall) is more important.

Table 1

Comparison of algorithm performance on b^3 and MUC metrics.

Model	Metric	MUC	b^3
CNN	Precision	24.23	97.88
	Recall	12.45	84.99
	F1	16.44	92.11
BiLSTM (Single-pass)	Precision	56.91	95.94
	Recall	30.20	88.76
	F1	29.46	92.21
BiLSTM (multiple-passes)	Precision	56.36	93.13
	Recall	39.68	90.43
	F1	45.88	91.76
RoBERTa	Precision	27.39	91.22
	Recall	13.10	89.65
	F1	17.72	90.43
Decision tree	Precision	73.46	98.15
	Recall	29.08	88.14
	F1	41.67	92.87

6. Conclusions

In the work, the task of coreference resolution for Ukrainian language is considered. An application that uses decision trees to search for coreference objects is created: data was prepared for analysis, a decision tree was built, its parameters were modified to achieve higher results, the performance of the application was evaluated on b^3 and MUC metrics. Decision trees, thanks to the

possibility of graphical representation, make it easier to analyze the obtained results and explain how one or another result is obtained, which distinguishes them from other popular methods of data analysis, in particular neural networks.

The obtained results show a relatively high efficiency of the algorithm - the highest results were achieved on the b^3 metric and relatively high on the BiLSTM metric, which indicates the expediency of using decision trees to search for coreference objects in Ukrainian-language texts.

7. References

[1] S. Pogorilyy, P. Biletskyi, Usage of a graphics processor to accelerate coreference resolution while using the RoBERTa model., Scientific works of DonNTU, Series: Informatics, cybernetics and computer technology, 2022, pp. 4-9.

[2] Z. Dzunic, S. Momcilovic, B. Todorovic, M. Stankovic, Coreference Resolution Using Decision Trees, 2006 8th Seminar on Neural Network Applications in Electrical Engineering, 2006, pp. 109-114.

[3] E. Fernandes, C. Nogueira, R. Milidiú. Latent Trees for Coreference Resolution. Computational Linguistics 40(4), 2014, pp.801–835.

[4] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettelemoyer, Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2018, pp. 2227–2237.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed Representations of Words and Phrases and their Compositionality. Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013, pp. 3111–3119.

[6] Scikit-learn library. URL: https://scikit-learn.org/ .

[7] Graphviz library. URL: https://graphviz.org/ .

[8] UDpipe library. URL: https://lindat.mff.cuni.cz/services/udpipe/ .

[9] S.Telenyk, S. Pogorilyy, A. Kramov , The complex method of coreferent clusters detection based on a BiLSTM neural network, Knowledge Based Systems, 2021, pp. 205-210.

[10] A. Bagga, B. Baldwin, Algorithms for Scoring Coreference Chains, The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference, 1998, pp. 563-566.

[11] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, L. Hirschman, A Model-Theoretic Coreference Scoring Scheme, Proceedings of the 6th Conference on Message Understanding (MUC), 1995, pp. 45-52.