Propaganda Detection in Text Data Based on NLP and Machine Learning

Vitaliia-Anna Oliinyk[1], Victoria Vysotska[2[0000-0001-6417-3689]], Yevhen Burov[3[0000-0001-6124-3995]], Khrystyna Mykich[4[0000-0002-4324-2080]], Vitor Basto-Fernandes[5[0000-0003-4269-5114]]

[1-4]Lviv Polytechnic National University, Lviv, Ukraine
[5]University Institute of Lisbon, Lisbon, Portugal
vitaliia-anna.oliinyk.sa.2017@lpnu.ua[1],
Victoria.A.Vysotska@lpnu.ua[2]

Abstract. The goal of this research is to build a machine-learning model, as well as preliminary data process and feature extraction algorithms that would allow to successfully identify signs of propaganda in text data and to solve a binary classification task. The task is presented in two forms: article level propaganda detection and sentence level propaganda detection. The propaganda detection dataset for this level of task consists of 35 993 articles (including headlines) written in English. Each article is marked as either "propaganda" or "non-propaganda". The dataset also contains unique identifier for each article.

Keywords: Content, Text Data, Propaganda Detection, Data Classification, Machine Learning.

1 Introduction

Before we start feature extraction process, we need to perform a few particular operations on the data to clean and prepare it for the extraction. First, we need to convert every word in the dataset to lowercase so that in the process of vectorization two semantically identical words, one uppercase and one lowercase, would not considered as separate tokens. In order to do so, we perform the following transformation:

```
data['article'] = data['article'].apply(lambda x: "
".join(x.lower() for x in x.split()))
```

Data is presented in the form of text file that consist of tab-separated article content, assigned class and unique article identifier. After the file is loaded and the identifier attribute is deleted, we convert our dataset to pandas.DataFrame format.

As is clearly presented on the graph, data is not balanced and the non-propagandistic articles are in majority. To be more specific, data contains 31 972 non-propagandistic and 4 201 propagandistic articles. After this, we need to view whether the data is evenly distributed by category.

	article	label
0	Et tu, Rhody? A recent editorial in the Provi...	non-propaganda
1	A recent post in The Farmington Mirror — our t...	non-propaganda
2	President Donald Trump, as he often does while...	non-propaganda
3	February is Black History Month, and nothing l...	non-propaganda
4	The snow was so heavy, whipped up by gusting w...	non-propaganda
5	Four months after the Sandy Hook School shooti...	non-propaganda
6	The first major newspaper article about Donald...	non-propaganda
7	For three years, starting in 2008, New York ar...	non-propaganda
8	President Donald Trump's tumultuous administra...	non-propaganda
9	With Hartford on edge about the future of Aetn...	non-propaganda
10	An employee at a Hibachi Express in Florida ha...	non-propaganda
11	With the toll of the carnage from the country'...	non-propaganda
12	The State Department's point-man on North Kore...	non-propaganda
13	The Trump Organization announced Monday that i...	non-propaganda
14	Aer Lingus' service from Bradley International...	non-propaganda
15	For its show "Constellations," which ends its ...	non-propaganda
16	The Corporation for Public Broadcasting (CPB) ...	non-propaganda
17	All five members of New Britain's state legisl...	non-propaganda
18	The leader and second in command of a credit-c...	non-propaganda

Fig. 1. Article-level dataset in pandas.DataFrame format.

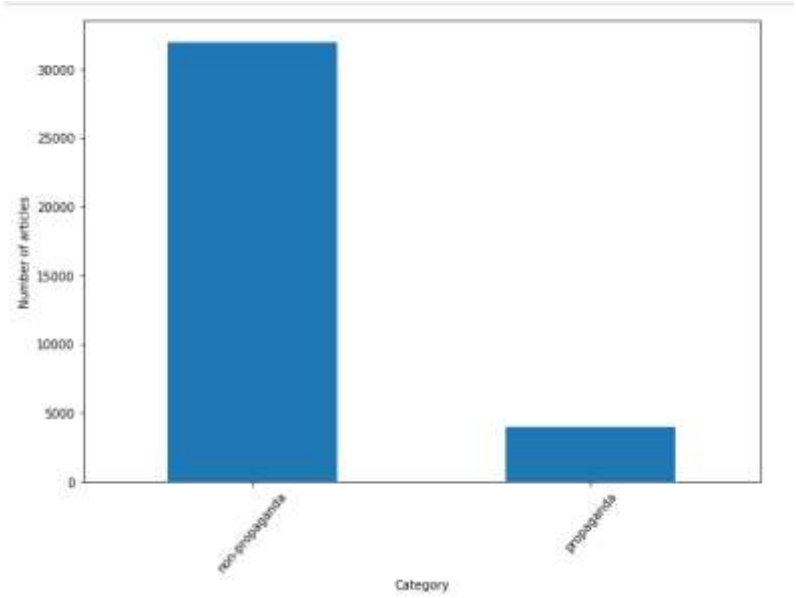

Fig. 2. Data distribution by category.

The next step is to remove punctuation symbols from the dataset, since on the task level it would not be a very informative feature, but it will also be counted as a separated token during vectorization process which can lead to data being noisy.

In order to do so, we perform the following transformation:

```
data['article'] = data['article'].str.replace('[^\w\s]','')
```

Lastly, we need to extract and remove so-called stop words. Stop words are usually the most common words in a language that do not contribute anything to the data semantically. Moreover, so will be no use for us in the process of building a model. We are using nltk library in-built corpora and remove them from the data in a loop.

```
stop = nltk.stopwords.words('english')
data['article'] = data['article'].apply(lambda x: " ".join(x for
x in x.split() if x not in stop))
```

After this, we can split our data into separate training and test sets.

```
X = data['article']
y = data['label']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

2 Feature Extraction

The raw data by itself does not carry any useful attributes and is not suitable for applying them in the machine-learning model, so we need to extract those features before we can run any machine-learning algorithm.

The first step in this process is text vectorization.

During the vectorization process, every word (term) in the corpus is assigned a unique number. Text data transforms into N-dimensional vector, where N is a number of words in the corpus. The value of each vector element is a term frequency of the corresponding word. We are going to use CountVectorizer class from scikit-learn library to implement text vectorization.

```
vectorizer = CountVectorizer(analyzer='word',
token_pattern=r'\w{1,}', ngram_range=(1,2),
strip_accents='unicode', min_df=3, max_df=0.5)
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

The return value of CountVectorizer fit and transform methods is a sparse matrix, the size of which equals to a number of unique words in corpus.

We need to make sure that, after the vectorization is applied, the number of features of training set and the number of features of test set are equal.

```
: X_train.shape
: (28794, 605048)

: X_test.shape
: (7199, 605048)
```

Fig. 3. Checking the training and test set dimensions.

The nest step of our feature extraction process is TF-IDF transformation [1-5].

TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

TF is a ratio of a raw count of a term in a document (the number of times the word t occurs in a document d) and the overall number of words in a document [6-9]. TF evaluates the importance of a particular word t_i in a document scope [10-14].

$$TF = \frac{n_i}{\sum_k n_k},$$

where n_i is the number of occurrences of the word n, while the denominator represents the overall number of words in a document [15-21].

$$IDF = \log \frac{|D|}{|(d \in D : t \in d)|},$$

where $|D|$ is the number of documents in the collection, while $|d \in D : t \in d|$ represents the number of documents that contain the word t [22-27].

$$TFIDF = TF \times IDF.$$

We will use TfidfTransofrmer class from scikit-learn library to implement TF-IDF transformation.

```
transformer = TfidfTransformer(use_idf=True, smooth_idf = True)
X_train = transformer.fit_transform(X_train)
X_test = transformer.transform(X_test)
```

The transformation is performed on the previously formed sparse matrix of vectorized text data.

3 Building a Model

For this classification task, we will use the logistic regression model [26-39].

Logistic regression uses logistic function to model binary dependent variable. It can described with the following mathematical equation:

$$P = \frac{e^{a+bX}}{1 + e^{a+bX}}$$

where P is a dependent variable that varies in the scope of $[0, 1]$, while X is a matrix of independent variables, and a i b are numeric coefficients of logistic model.

We will use LogisticRegression class from scikit-learn library to implement logistic regression model.

```
model = LogisticRegression(penalty='l2',
class_weight='balanced', solver='lbfgs')
model.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Fig. 4. Logistic regression model.

The parameters specified are: penalty='l2', which indicates that for the purpose of regularization our model will use Ridge regression approach, while the parameter solver='lbfgs' indicates that for the purpose of optimization our model will be using the limited memory Broyden–Fletcher–Goldfarb–Shanno algorithm.

4 Model Evaluation

To evaluate our model, we will use it to predict values for the test set data.

```
predictions = model.predict(X_test)
```

Then we'll build a confusion matrix.

```
n = len(conf_matrix)
plt.figure(figsize = (n, n))
sns.heatmap(conf_matrix, annot=True)
plt.title("Confusion Matrix")
plt.ylabel("Actual Value")
plt.xlabel("Predicted Value")
plt.show()
```

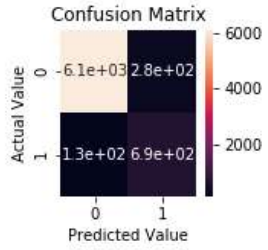

Fig. 5. Article level model confusion matrix

The interpretation of the confusion matrix indicates that our model has successfully classified 6097 non-propaganda articles & 694 propaganda articles, but failed to classify 123 propaganda articles and 285 non-propaganda articles.

To view the model score:

```
model.score(X_test, y_test)
```

The achieved model score equals to 0.9433254618697041.

5 Sentence Level Propaganda Detection Task

The dataset for this type of task contains approximately 540 articles in English, including headlines. The articles are broken up into separate sentences. Each sentence is labelled as either “propaganda” or “non-propaganda”. The dataset also includes unique identifier for each article, as well as a unique identifier for each sentence on the article scope. The dataset contains a total of 14263 sentences.

Data is stored in a form of text file collection, with a separate file for each article. The data attributes are also stored separately. After each file collection is loaded, we concatenate them and form a single pandas.DataFrame, while also elimination articles and sentences unique identifiers from the data.

	sentence	label
0	US bloggers banned from entering UK	non-propaganda
2	Two prominent US bloggers have been banned fro...	non-propaganda
4	Pamela Geller and Robert Spencer co-founded an...	propaganda
6	They were due to speak at an English Defence L...	non-propaganda
8	A government spokesman said individuals whose ...	non-propaganda
...	...	...
15164	This is a Moon of Alabama fundraiser week.	non-propaganda
15165	No one pays me to write these blog posts.	non-propaganda
15166	If you appreciated this one, or any of the 7,0...	non-propaganda
15167	Posted by b on November 29, 2018 at 10:23 AM |...	non-propaganda
15168	Comments	non-propaganda

14263 rows × 2 columns

Fig. 6. The sentence level dataset.

The next step is to once more build a plot to see how our data is distributed in the terms of category and to evaluate the level of balance in the data.

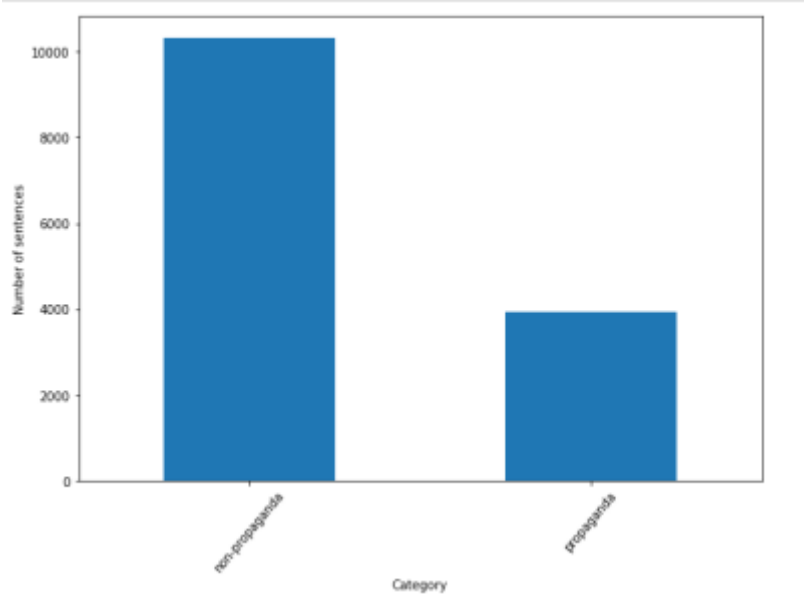

Fig. 7. Sentence-level data distribution according to category.

As is evident from the plot, the data is once again not evenly distributed and unbalanced. The number of non-propaganda articles outweighs propaganda articles in this dataset as well. Specifically, the dataset contains 10 325 non-propaganda articles and 3938 propaganda articles. First, we repeat the approach we have taken while solving the previous task and remove all stop words from the data.

```
stop = stopwords.words('english')
data['sentence'] = data['sentence'].apply(lambda x: " ".join(x
for x in x.split() if x not in stop))
```

After that, we convert our text data to lowercase once again.

```
data['sentence'] = data['sentence'].apply(lambda x: "
".join(x.lower() for x in x.split()))
```

We do not eliminate punctuation symbols this time, since we will need them at during a feature extraction process. At the beginning of this process, we perform a POS (Part-of-Speech) tagging operation. In order to do so, we use the spacy library.

```
nlp = spacy.load('en')
def tag(sentence):
    global nlp
    doc = nlp(sentence)
    return " ".join([f'{x.text}_{x.tag_}_{x.lemma_}' for x in
doc])
sentences_pos = copy.deepcopy(data['sentence'])
```

```
tagged = pos_tagging(sentences_pos)
data['tagged'] = tagged
```

After this operation, our data has a new column called “tagged” that presents every word in a sentence in the following format: “%word%_%part of speech tag%_%lemma%”.

The next step is to perform a few manual feature extraction technique.

Firstly, we need to mark every sentence as either containing or not-containing quotations to identify the “Appeal to authority” propaganda technique.

In order to do so, we initialize the corresponding function.

```
def get_quotations(sentences):
    result = []
    for sentence in sentences:
        match = 1 if '"' in sentence else 0
        result.append(match)
    return np.array(result).reshape(-1, 1)
```

The next step is to check each sentence for the presence of so-called glitter words. These are the words like “patriotism”, “democracy”, “duty”, “power” etc. We do this to identify “Slogan” and “Flag Waving” propaganda techniques.

In order to do so, we compute the number of matches of words from a sentence with words from a glitter word lexicon and divide it by the overall number of words in a sentence in order to normalize the coefficient. We form a separate text file (a lexicon) with such words and initialize the corresponding function.

```
def get_glitter(tagged):
    filename = 'glitter_words.txt'
    glitters = []
    append = glitters.append
    with open(os.path.join(LEXICONS_PATH, filename),
encoding='utf-8') as f:
        for line in f.readlines():
            append(line.replace('\n', ''))
    result = []
    for sentence in tagged:
        words = 0
        matches = 0
        for wline in sentence.split():
            try:
                w, t, l = wline.split("_")
            except:
                continue
            w = w.lower()
            l = l.lower()
            words+=1
```

```
            if l in glitters or w in glitters:
                matches+=1
        if words == 0:
            result.append(0)
        else:
            result.append(matches/words)
    return np.array(result).reshape(-1, 1)
```

Then, with a similar approach, we check each sentence for the presence of intensifying words (“absolute”, “total”, “very”, “incredible” etc.) and absolute pronouns (“everyone”, “nobody” etc.). While doing so, we try to identify such propaganda techniques as “Loaded language” and “Bandwagon”.

We form similar lexicons for each task and initialize similar functions. Finally, we victories our text data using Word2Vec shallow two-layer neural net with pre-trained Twitter 200-dimensional pre-trained model to perform word embedding’s.

In order to do so, we will use gensim library.

```
w2v_file = os.path.join(WORD2VEC_PATH, 'twitter.27B.200d.txt')
w2v_model = KeyedVectors.load_word2vec_format(w2v_file,
binary=False)
def w2v_vectorize(tagged):
    global w2v_model
    X = []
    ndims = 200
    for sentence in tagged:
        words = []
        for wline in sentence.split():
            try:
                w, t, l = wline.split("_")
            except:
                continue
            words.append(w)
        row_data = np.mean([w2v_model[w] for w in words if w in
w2v_model] or
                           [np.zeros(ndims)], axis=0).tolist()
        X.append(row_data)
    X = np.array(X)
    X_std = (X - X.min(axis=0)) / (X.max(axis=0) -
X.min(axis=0))
    X_scaled = X_std * (1 - 0) + 0
    return X_scaled
```

Finally, we create a new pandas.DataFrame on the basis of the already existing one with the use of all aforementioned operations.

```
word2vec_features = w2v_vectorize(data['tagged'])
word2vec_columns = [f'dim{x}' for x in range(200)]
```

```
glitter_words = get_glitter(data['tagged'])
quotations = get_quotations(data['sentence'])
intensifiers = get_intensifiers(data['tagged'])
absolutes = get_absolutes(data['tagged'])
X = pd.DataFrame(word2vec_features, columns=[word2vec_columns])
X['quotations'] = quotations
X['glitter_words'] = glitter_words
X['intensifiers'] = intensifiers
X['absolutes'] = absolutes
y = data['label']
```

Split dataset into training and test sets.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

We will also use Logistic regression model for this task, as well as Grid Seacrh cross-validation algorithm to find best parameters of our model.

```
lr_model = LogisticRegression()
penalty = ['l1', 'l2']
C = np.logspace(0, 4, 10)
hyperparameters = dict(C=C, penalty=penalty)
clf = GridSearchCV(lr_model, hyperparameters, refit='f1', cv=5)
```

Best model parameters: penalty='l2', C=7.74.

To evaluate the efficiency of our model, we predict values on test set.

```
predictions = best_model.predict(X_test)
```

Build confusion matrix.

```
n = len(conf_matrix)
plt.figure(figsize = (n, n))
sns.heatmap(conf_matrix, annot=True)
plt.title("Confusion Matrix")
plt.ylabel("Actual Value")
plt.xlabel("Predicted Value")
plt.show()
```

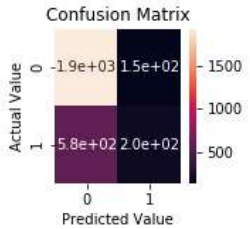

Fig. 8. Confusion matrix.

The confusion matrix can be interpreted in the following way: our model has successfully classified 1917 non-propaganda articles and 205 propaganda articles, but 585 propaganda articles and 146 non-propaganda articles were misclassified.

We can also view model score:

```
best_model.score(X_test, y_test)
```

Model score equals to 0.7437784787942516.

6 Conclusions

During the research conduction, two-machine learning models are built to identify propaganda – one for article level task and one for sentence level task. In the process vectorization, TF-IDF, POS-tagging, word embedding and manual feature extraction techniques are applied. Classification has performed with the use of Logistic Regression model and Grid Search cross-validation algorithm. The model received following scores: 0.94 for article-level model and 0.74 for sentence level model.

References

1. Propaganda definitions, https://propaganda.qcri.org/annotations/definitions.html.
2. Text Summarization using NLTK: TF-IDF Algorithm, https://towardsdatascience.com/text-summarization-using-tf-idf-e64a0644ace3.
3. Spacy: Linguistic Features, https://spacy.io/usage/linguistic-features.
4. Introduction to Word Embedding and Word2Vec, https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa.
5. Glittering Generalities, [https://marketingwit.com/examples-of-glittering-generalities.
6. Lytvyn, V., Vysotska, V., Budz, I., Pelekh, Y., Sokulska, N., Kovalchuk, R., Dzyubyk, L., Tereshchuk, O., Komar, M.: Development of the quantitative method for automated text content authorship attribution based on the statistical analysis of N-grams distribution. In: Eastern-European Journal of Enterprise Technologies, 6(2-102), pp. 28-51. (2019).
7. Lytvyn, V., Vysotska, V., Hamon, T., Grabar, N., Sharonova, N., Cherednichenko, O., Kanishcheva, O.: Preface: Computational Linguistics and Intelligent Systems (COLINS-2020). In: CEUR Workshop Proceedings, Vol-2604. (2020)
8. Bisikalo, O., Vysotska, V., Burov, Y., Kravets, P.: Conceptual Model of Process Formation for the Semantics of Sentence in Natural Language. In: CEUR workshop proceedings, Vol-2604, 151-177. (2020)
9. Vysotska, V.: Ukrainian Participles Formation by the Generative Grammars Use. In: CEUR workshop proceedings, Vol-2604, 407-427. (2020)
10. Bisikalo, O., Vysotska, V.: Linguistic analysis method of Ukrainian commercial textual content for data mining. In: CEUR Workshop Proceedings, Vol-2608, 224-244. (2020)
11. Khairova, N., Kolesnyk, A., Mamyrbayev, O., Petrasova, S.: Applying VSM to Identify the Criminal Meaning of Texts. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 20-31. (2020).
12. Lande, D., Dmytrenko, O., Radziievska, O.: Subject Domain Models of Jurisprudence According to Google Scholar Scientometrics Data. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 32-43. (2020).

13. Bisikalo, O., Kontsevoi, A.: System for Definition of Indicator Characteristics of Social Networks Participants Profiles. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 77-88. (2020).
14. Levchenko, O., Tyshchenko, O., Dilai, M.: Associative Verbal Network of the Conceptual Domain БІДА (MISERY) in Ukrainian. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 106-120. (2020).
15. Vasyliuk, V., Shyika, Y., Shestakevych, T.: Information System of Psycholinguistic Text Analysis. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 178-188. (2020).
16. Khomytska, I., Teslyuk, V.: The Multifactor Method Applied for Authorship Attribution on the Phonological Level. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 189-198. (2020).
17. Andrunyk, V., Berko, A., Rusyn, B., Pohreliuk, L., Chyrun, S., Dokhniak, B., Karpov, I., Krylyshyn, M.: Information System of Photostock Web Galleries Based on Machine Learning Technology. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 1032-1059. (2020).
18. Zdebskyi, P., Lytvyn, V., Burov, Y., Rybchak, Z., Kravets, P., Lozynska, O., Holoshchuk, R., Kubinska, S., Dmytriv, A.: Intelligent System for Semantically Similar Sentences Identification and Generation Based on Machine Learning Methods. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 317-346. (2020).
19. Andrunyk, V., Vasevych, A., Chyrun, L., Chernovol, N., Antonyuk, N., Gozhyj, A., Gozhyj, V., Kalinina, I., Korobchynskyi, M.: Development of Information System for Aggregation and Ranking of News Taking into Account the User Needs. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 1127-1171. (2020).
20. Makara, S., Chyrun, L., Burov, Y., Rybchak, Z., Peleshchak, I., Peleshchak, R., Holoshchuk, R., Kubinska, S., Dmytriv, A.: An Intelligent System for Generating End-User Symptom Recommendations Based on Machine Learning Technology. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 844-883. (2020).
21. Husak, V., Lozynska, O., Karpov, I., Peleshchak, I., Chyrun, S., Vysotskyi, A.: Information System for Recommendation List Formation of Clothes Style Image Selection According to User's Needs Based on NLP and Chatbots. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 788-818. (2020).
22. Lytvyn, V., Kubinska, S., Berko, A., Shestakevych, T., Demkiv, L., Shcherbyna, Y.: Peculiarities of Generation of Semantics of Natural Language Speech by Helping Unlimited and Context-Dependent Grammar. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 536-551. (2020).
23. Bekesh, R., Chyrun, L., Kravets, P., Demchuk, A., Matseliukh, Y., Batiuk, T., Peleshchak, I., Bigun, R., Maiba, I.: Structural Modeling of Technical Text Analysis and Synthesis Processes. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 562-589. (2020).
24. Chyrun, L.: Model of Adaptive Language Synthesis Based On Cosine Conversion Furies with the Use of Continuous Fractions. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 600-611. (2020).
25. Perkhach, R., Kysil, D., Dosyn, D., Zavuschak, I., Kis Y., Hrendus, M., Vasyliuk, A., Sadova, M., Prodanyuk, M.: Method of Structural Semantic Analysis of Dental Terms in the

Instructions for Medical Preparations. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 662-669. (2020).
26. Khomytska, I., Teslyuk, V., Holovatyy, A., Morushko, O.: Development of methods, models, and means for the author attribution of a text. In: Eastern-European Journal of Enterprise Technologies, 3(2-93), 41–46. (2018)
27. Khomytska, I., Teslyuk, V.: Authorship and Style Attribution by Statistical Methods of Style Differentiation on the Phonological Level. In: Advances in Intelligent Systems and Computing III. AISC 871, Springer, 105–118. (2019)
28. Stasiuk, L.: Computer Sampling and Quantitative Analysis in Exploring Secondary Functions of Questions in Speech Genres of Intimate Communication. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 227-238. (2020).
29. Hadzalo, A.: Analysis of Gender-Marked Units: Statistical Approach. In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 462-471. (2020).
30. Romanyshyn, N.: Application of Corpus Technologies in Conceptual Studies (based on the Concept Ukraine Actualization in English and Ukrainian Political Media Discourse). In: Computational Linguistics and Intelligent Systems, COLINS, CEUR workshop proceedings, Vol-2604, 472-488. (2020).
31. Saaya, Z., & Hong, T. (2019).: The development of trust matrix for recognizing reliable content in social media. International Journal of Computing, 18(1), 60-66.
32. Zhezhnych, P., Shilinh, A., & Melnyk, V.: Linguistic analysis of user motivations of information content for university entrant's web-forum. International Journal of Computing, 18(1), 67-74. (2019).
33. Pach, J., Bilski, P.: A robust binarization and text line detection in historical handwritten documents analysis. In: International Journal of Computing, 15(3), 154-161 (2016).
34. Batura, T., Bakiyeva, A., Charintseva, M.: A method for automatic text summarization based on rhetorical analysis and topic modeling. In: International Journal of Computing, 19(1), 118-127. (2020).
35. Sachenko, S., Pushkar, M., Rippa, S.: Intellectualization of Accounting System. In: International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applicationsm, 536 – 538. (2007)
36. Sachenko, S., Rippa, S., Krupka, Y.: Pre-Conditions of Ontological Approaches Application for Knowledge Management in Accounting. In: International Workshop on Antelligent Data Acquisition and Advanced Computing Systems: Technology and Applications,. 605-608. (2009)
37. Babichev, S.: An Evaluation of the Information Technology of Gene Expression Profiles Processing Stability for Different Levels of Noise Components. In: Data, 3 (4), art. no. 48. (2018)
38. Babichev, S., Durnyak, B., Pikh, I., Senkivskyy, V.: An Evaluation of the Objective Clustering Inductive Technology Effectiveness Implemented Using Density-Based and Agglomerative Hierarchical Clustering Algorithms. In: Advances in Intelligent Systems and Computing, 1020, 532-553. (2020)
39. Yurynets, R., Yurynets, Z., Dosyn, D., Kis, Y.: Risk Assessment Technology of Crediting with the Use of Logistic Regression Model. In: CEUR Workshop Proceedings, Vol-2362, 153-162. (2019)