MONOGRAPHIC SERIES
«EUROPEAN SCIENCE»

BOOK 41
PART 2

SWorld
Germany

Dumyn I., Basystiuk O., Hymon S.

ENTWICKLUNG DES WISSENSCHAFTLICHEN DENKENS

METHODEN UND WERKZEUGE ZUR INTELLIGENTEN ANALYSE UKRAINISCHSPRACHIGER TEXTE ZUR IDENTIFIZIERUNG EMOTIONALER MARKER UND ZUR BESTIMMUNG DER BETONUNG IM TEXT

SCIENTIFIC THOUGHT DEVELOPMENT

METHODS AND TOOLS FOR INTELLIGENT ANALYSIS OF UKRAINIAN-LANGUAGE TEXTS TO IDENTIFY EMOTIONAL MARKERS AND DETERMINE STRESS IN THE TEXT

Monographic series «European Science»
Book 42. Part 2.

In internationalen wissenschaftlich-geometrischen Datenbanken enthalten
Included in International scientometric databases

MONOGRAPHIE
MONOGRAPH

ScientificWorld-NetAkhatAV
Karlsruhe 2025

Authors:
Dumyn I., Basystiuk O., Hymon S.

Entwicklung des wissenschaftlichen Denkens: Methoden und Werkzeuge zur intelligenten Analyse ukrainischsprachiger Texte zur Identifizierung emotionaler Marker und zur Bestimmung der Betonung im Text. Monografische Reihe «Europäische Wissenschaft». Buch 42. Teil 2. 2025.

Scientific thought development: Methods and tools for intelligent analysis of ukrainian-language texts to identify emotional markers and determine stress in the text. Monographic series «European Science». Book 42. Part 2. 2025.

ISBN 978-3-98924-098-8
DOI: 10.30890/2709-2313.2025-42-02

Published by:
ScientificWorld-NetAkhatAV
Lußstr. 13
76227 Karlsruhe, Germany
e-mail: *editor@promonograph.org*
site: https://desymp.promonograph.org

ÜBER DIE AUTOREN / ABOUT THE AUTHORS

1. *Dumyn Iryna,* Candidate of Technical Sciences, LPNU, ORCID 0000-0001-5569-2647
2. *Basystiuk Oleh,* LPNU, ORCID 0000-0003-0064-6584
3. *Hymon Sofia,* LPNU

Inhalt / Content

CHAPTER 5
SYSTEMS AND TOOLS DEVELOPMENT

INTRODUCTION

In today's information society, the rapid growth of text data, particularly on social networks, news resources, psychological support forums, and chatbots, creates an objective need for highly accurate tools for the automatic analysis of the emotional content of speech. This issue is particularly relevant in connection with the need for timely detection of signs of psychological stress, anxiety, depression, or other affective states based on users' written discourse. In this context, the effective identification of emotional markers in Ukrainian-language texts is not only a linguistic task, but also an interdisciplinary one at the intersection of computational linguistics, psycholinguistics, artificial intelligence, and social psychology.

The relevance of the topic is due to the need to create a methodological basis and applied solutions for the intellectual analysis of Ukrainian-language texts to identify manifestations of stress. Despite the active development of natural language processing for English-language sources, the processing of the Ukrainian language, particularly in the emotional and psycho-emotional dimensions, remains under-researched. Given the challenges associated with war, social instability, and information overload, the task of timely detection of stressful communications is of not only scientific but also social importance.

The aim of the study is to develop, justify, and experimentally test methods and means of automatic analysis of texts in Ukrainian to identify emotional markers that signal the presence or likelihood of the author's stressful state.

Research objectives:

– to formulate the theoretical foundations of emotional text analysis and stress detection based on linguistic and psycholinguistic features;

– to analyze existing methods of emotion detection, tone classification [56], and thematic modeling;

– to adapt modern approaches to Ukrainian-language corpora, considering syntagmatic and paradigmatic connections;

– to develop an approach to building a corpus of annotated texts with classification

[61] based on stress.

The object of research is text messages in Ukrainian in social, informational, psychological, and everyday contexts.

The subject of research is methods for the automatic detection of emotional markers and stress indicators in Ukrainian texts based on linguistic, statistical, and neural network models.

The scientific novelty lies in a comprehensive approach to the detection of stress states in Ukrainian-language texts, which combines analysis of morphosyntactic and word-formation structures; consideration of syntagmatic and paradigmatic relationships; the construction of contextually dependent representations of words and sentences; the use of transformer models for modeling emotional dependence in texts. For the first time, an adapted system of multilevel annotation of emotional states has been proposed, considering the thematic and lexical-grammatical context.

The practical significance of the research lies in the creation of a theoretical basis for tools that can be used to automatically detect the psycho-emotional load of texts in areas such as psychological monitoring, educational technologies, support services, media analysis, content moderation, as well as in the development of language models and chatbots with emotional sensitivity. The proposed models and approaches can be integrated into information and analytical systems for assessing risks related to the mental health of users.

KAPITEL 1 / *CHAPTER 1*
THEORETICAL FOUNDATIONS OF INTELLECTUAL ANALYSIS OF TEXTS

1.1. Basic concepts

Emotion is a complex psychophysiological process that reflects a person's subjective reaction to internal or external stimuli. It is accompanied by changes in behavior, language, facial expressions, and physiology. In the text [33], emotions manifest themselves as semantic [23], grammatical, stylistic, or pragmatic signals [20].

The classification [10-11] of emotions can be based on different models:

- basic emotions according to P. Ekman − joy, sadness, fear, anger, disgust, surprise;
- R. Plutchik's circle of emotions model: 8 basic emotions that combine into a more complex palette;
- A two-dimensional model based on valence and activation allows emotions to be placed on a scale of "pleasant/unpleasant" and "calm/excited".

In natural language processing, emotions are analyzed using lexical markers, machine learning models, or contextual language models [84].

Emotional markers are linguistic elements of a text that signal the emotional state of the author or character. They can be expressed at different linguistic levels:

- lexical level − words with a pronounced emotional coloring (e.g. happy, angry, scared);
- morphological level − diminutive and affectionate suffixes (e.g., girl, sweetheart);
- syntactic level − exclamations, rhetorical questions, short phrases, repetitions;
- punctuation level − frequent use of exclamation marks, dashes, and quotation marks.

Emotional markers can also be implicit (hidden in context or style) or explicit (expressed directly). For example, "I'm scared" is an explicit marker or "I heard

something creaking outside the window…" is an implicit marker of fear.

Within the framework of automatic analysis, markers help the model determine the emotional context of the text.

Stress in the text is an indicator of psychological tension or anxiety, manifested through specific language structures. Stress is not an emotion in the classical sense but is accompanied by a complex of emotional states (anxiety, irritation, fear) [21]. The main linguistic signs of stress in the text are a high frequency of negatively colored words, violation of grammatical structure, discontinuous speech, frequent use of words to denote danger, uncertainty (never, don't know, scary, end, tired), an increase in the frequency of first-person pronouns (I, me, mine), which indicates immersion in one's own state

Automatic stress detection requires not only word analysis, but also considering context, topic field, and emotional dynamics.

Sentiment is the overall emotional assessment that the author expresses about a particular object or situation. It can be:

- positive, expresses approval, satisfaction, support;
- negative, expresses dissatisfaction, fear, disappointment;
- neutral, description without emotional evaluation.

Determining tone is a key step in analyzing sentiment on social media, reviews, news, etc.

Tone is closely related to emotions but is a simplified representation of them. Table 1 shows classification of basic emotions [64], their brief description and corresponding tonality.

Below is the table 2 with emotions that often signal tension, anxiety, psycho-emotional exhaustion and are key markers in automatic stress detection systems.

In a research model, these emotions should be labeled as target classes for stress detection and used as a lexical basis for a dictionary or annotation for fine-tuning the model [31]. Analyzed in the context of syntactic/semantic patterns that accompany these emotions.

Table 1 - Classification of basic emotions and their corresponding tonality

Emotion	Description	Tonality
Joy	A state of pleasure, happiness, elation	Positive
Surprise	Reaction to an unexpected event or information	Neutral / ±
Sadness	A state of depression, loss, regret	Negative
Fear	Feelings of threat, danger, anxiety	Negative
Anger	Aggression, indignation, dissatisfaction	Negative
Disgust	Disgust, rejection, moral rejection	Negative
Trust	A feeling of security, faith in someone or something	Positive
Expectation	Focus on a future event (can be positive or negative)	Neutral / ±
In love / love	Deep affection, empathy, tenderness	Positive
Hatred	Intense negative emotion directed at an object	Negative
Sarcasm / irony	A hidden emotion, often expressed through an inversion of meaning	Mixed / contextual
Rest	A state of balance, inner harmony	Neutral / positive

Table 2 - List of basic emotions with descriptions and their corresponding interpretation in terms of the level of stress caused

Emotion	Description	Interpretation regarding stress
Fear	Reaction to a real or perceived threat	Immediate marker of anxiety, threat
Anxiety	Constant internal tension, anticipation of danger	Chronic stress, anxiety
Sadness	Feelings of loss, low mood	A sign of emotional exhaustion
Despair	Feelings of hopelessness, emotional collapse	High level of psycho-emotional pressure
Anger/Rage	Reaction to frustration, internal conflict	Associated with nervous tension
Frustration	Feeling of dissatisfaction, inability to achieve what you want	Often provoked or accompanied by stress
Despondency	Loss of faith in oneself, the situation, or others	Concomitant condition with prolonged stress
Tension	Feeling of psychological or physiological tightness	Direct symptom of stress

Perplexity	Loss of orientation, inability to make decisions	Often occurs with cognitive overload
Panic	Acute form of fear with loss of self-control	Peak stress response
Fatigue (emotional)	Exhaustion, apathy, loss of interest	A sign of prolonged stress
Fault	Feeling of personal responsibility for a negative outcome	May worsen stress
Shame	Internal tension due to judging oneself as "inadequate"	Decreased self-esteem, prolonged pressure

Stress is manifested as a specific form of the emotional state of the author of the text based on the formed emotional markers that are expressed in the text and allow to detect stress or positive mood as a generalized result of the analysis of the emotional context of the text. These concepts form the basis for building systems for automatic emotional analysis of Ukrainian-language texts in the context of detecting the psycho-emotional state of a person.

1.2. Linguistic aspects of Ukrainian text analysis and emotional text perception

Paradigmatic and syntagmatic relations are considered as the basic linguistic relations that describe the complex structure of a language system. This distinction applies to all levels of description. It was introduced by the Swiss linguist Ferdinand de Saussure in 1916 as a generalization of the traditional concepts of paradigm and syntagma.

A paradigm (Greek: parádeigma "sample, model") is a set of homogeneous forms opposed to each other on semantic and formal grounds.

A syntagma (Greek: syntagma, "that which is gathered in order") is a structured syntactic sequence of language elements formed by segmentation, which may consist of sounds, words, phrases, clauses, or entire sentences.

Paradigmatic relations exist between units of the language system outside the lines where they occur together. They are based on the criteria for the selection and

distribution of language elements. Paradigmatic relations that define the lexical system are based on the interdependence of words in the lexical composition: synonyms, antonyms, hyponymy, meronymy. Paradigmatic relations are vertical.

F. de Saussure called paradigmatic relations as associative relations because they represent the connection between individual elements in a specific environment.

It was the Danish linguist Louis Helmslev who replaced the term associative relations with paradigmatic relations.

Syntagmatic relations are direct linear connections between units of a segmental sequence. Syntagmatic relations are horizontal because they are based on the linear nature of speech.

Working principles

A syntagmatic structure is a set of words grouped around a nucleus with the same syntactic and semantic function. It is any syntactic structure consisting of one or more words. The combination occurs in the presence of other linguistic elements.

Paradigmatic structure is a vertical connection of signs due to the absence of other linguistic elements. They are continuously in a paradigmatic relationship of all inheritances of a verbal radical, where one appears, it can appear, replacing it, by any other from the verbal paradigm.

For the linguistic analysis of the emotional expressiveness of a text, it is important to consider both syntagmatic and paradigmatic relations between lexical units. Below are examples demonstrating both types of relations for sentences in Ukrainian with expressed emotions.

Example 1. The emotion of anxiety. Consider the sentence “I can’t sleep all night, anxiety is eating me up from the inside.”. Since syntagmatic connections are linear and horizontal, the sequential arrangement of words in the sentence “I can’t sleep” to “anxiety is eating up” ensures grammatical coherence and semantic integrity of the statement. The elements are combined according to the rules of subject, predicate, and complement syntax. The structure reflects the causality between the emotional state and the physiological effect. Within the paradigmatic axis, lexical substitutions are possible without violating the grammatical structure, for example, the word “anxiety”

can be replaced by “panic”, “tension”, which belong to the fear paradigm, which confirms the associativity and verticality of paradigmatic connections. Similarly, the verb “eats up” can be replaced with the words “exhausts”, “presses”, “oppresses”, which intensifies or modifies the emotional tone.

Example 2. The emotion of joy. In the sentence “When I saw the dawn in the mountains, my soul was filled with happiness.” the structure includes a complex construction with a temporal subordinate clause “when I saw”, which is logically combined with the main clause “my soul was filled with happiness”. The elements are interconnected functionally and sequentially as for syntagmatic connections: observation causes emotional reaction. The keyword “happiness” can be replaced by synonymous units “joy”, “delight”, “inspiration”, which creates a paradigm of emotions of a positive spectrum. The verb “filled” has several functional equivalents, such as “overflowed”, “filled”, “exploded”, which change the intensity of emotional expression.

For paradigmatic relations, all symbols (e.g., words) are considered paradigms, which are members of a certain class or semantic group. Paradigms are said to establish a paradigmatic relation – or associative relation – if they can be interchanged, at least without changing the grammatical coherence of the sentence. Therefore, a paradigm is usually considered to be a set of symbols that belong to a certain class, e.g., verbs or nouns are examples of grammatical paradigms.

So, the paradigmatic view considers a sentence as a sequence of disjunctions of paradigms $\mathbf{p_0 \vee p_1 \vee p_2 \vee p_3 \ldots \vee p_i}$ where $\mathbf{p_i}$ — a specific paradigm. As an example, in the example above the word “happiness” can be replaced by the word “joy” without affecting the grammatical coherence of the sentence. It can also refer to the lexical category of a syntagm or even its meaning, when the classes refer to synonyms, antonyms, etc.

Syntagmatic relations correspond to a chain of associations of symbols that make up a sentence or a larger lexical unit. The choice of specific paradigms and their combination generates a syntagma, which conveys a sequence of lexical units connected by syntagmatic relations.

This helps to determine the content of the sentence. The syntagma is regulated by the grammar of the language used, that is, by the rules that determine the coherence of the expression of a certain chain of symbols. According to the syntagmatic view, the sentence is considered as a sequence of combinations of paradigms of the form $\mathbf{p_0 \wedge p_1 \wedge p_2 \wedge p_3 \ldots \wedge p_i}$ with $\mathbf{p_i}$ as a particular paradigm, e.g. the two sentences presented in the table above correspond to specific chains of paradigms.

Paradigmatic analysis is concerned with the study of specific patterns in texts, in contrast to syntagmatic analysis, which is based on the analysis of the surface representation of language, in which the word is the central component. It is worth noting the importance of these two types of structural relations for the discussion of word relationships. However, due to the different nature of these relations and the different information they convey, some scholars have proposed a distinction between paradigmatic and syntagmatic relations.

Paradigmatic affinity refers to the notion of interchangeability with respect to the effect of replacing one word with another, e.g. on the grammatical coherence of a sentence. More precisely, the paradigmatic affinity of two words can be viewed as a function of the effect of replacing two words with the meaning conveyed by the sentence. Therefore, paradigmatic relations between words should be assessed using indirect relations. As an example, the two synonyms father and dad rarely occur together, but rather tend to occur with the same words – this can be captured by analyzing second-order co-occurrences or, more generally, indirect co-occurrences of words. Paradigmatic affinity is a broad concept. Indeed, the association of specific paradigms in the same semantic group can be justified by various semantic relations that can be established between paradigms, e.g. hypo/hypernymy, synonymy, antonymy.

Syntagmatic connection is mainly fixed within a certain text area through the co-location of words and direct, that is, first-order co-formation.

Paradigmatic and syntagmatic relations are among the central ones in defining semantic measures based on the corpus of language.

Paradigmatic relations in language reflect the dependencies between words that

can appear together in certain contexts. The main tasks of paradigmatic relations are to find common paradigmatic relations and to recognize paraphrases [89]. When analyzing a text, it is possible to identify words that often occur together in different contexts. Finding such relations can help to establish common themes in the text, as well as help in building speech models. As for paraphrases, these are different options for expressing the same thought, respectively, paradigmatic relations can be used to recognize paraphrases. For example, the word "автомобіль" can be replaced by the word "машина", since they have a common paradigmatics.

Syntagmatic relations can help you better understand how to use words correctly in a sentence to achieve clarity and accuracy in your expression.

The concept of context

The concept of context is one of the central concepts for the intellectual analysis of texts. This concept is of great importance for fixing the meaning of a word through the analysis of syntagmatic and paradigmatic relations. Indeed, the meaning of a word is usually considered to be understandable only without the context of use, that is, according to its use, which is determined through the structural relations that the word establishes. Thus, the concept of context is central for performing structural processing of natural language through the analysis of paradigmatic and syntagmatic relations. As an example, the co-occurrence of first-order words will be assessed by studying the syntagmatic relations between words.

Scholars have proposed several approaches to defining the context of a word. They differ in their linguistic complexity, algorithmic complexity, reliability, and the information they consider. The basic approach is to consider the context of a word as a corpus document in which the word occurs. This definition of context refers to the semantic model underlying the vector space model (VSM), a well-known model that was proposed in information retrieval to characterize the vocabulary of documents that make up a corpus. In VSM, a document is considered as a topic unit. It is represented as a vector that highlights the words it contains. To this end, the corpus is used to derive a semantic model that corresponds to a document-word matrix. A basic example of such a matrix is presented in Table 3 below – w_i refers to word i, and d_j to document j

of the corpus.

Table 3 – Example of a document-word matrix for creating a semantic model

	d_0	d_1	d_2	d_3
w_0	1	1	0	0
w_1	0	1	0	0
w_2	1	1	1	0
w_3	1	1	0	0

If the word w_i occurs in the document d_j, the cell (w_i, d_j) will be filled with the value 1. Based on this model, a query based on a vector representation will be used to find more relevant documents. The information retrieval system will be based on a similarity function that will evaluate the similarity between the vector representation of the query and the vector representation of each document (i.e., the columns of the matrix).

Thus, considering the transpose of the document-word matrix used in VSM, we can assume that each word is also represented according to a vector representation − the rows of the matrix in the table, for example, w_0 is represented by the vector [1, 1, 0, 0]. This vector representation of a word selects the documents in which this word occurs. Thus, given that vector similarity can be estimated using specific mathematical functions, the semantic relatedness of two words can be simply defined as a function of the similarity of their vector representations.

This general approach does not limit the words that are characterized by documents, but allows building semantic models in which words are analyzed relative to the context, for example, a paragraph, a sentence - in the case of VSM, the semantic model refers to the document-word matrix, so the chosen context is a document.

Thus, in a more general form, a semantic model can be defined as a matrix in which each row i is a vector representation of a word w_i, and each dimension of the vector refers to a specific context c_n, i.e., row i provides a vector representation of the corresponding word w_i with $\overrightarrow{w_i} = (c_1, c_2, \ldots, c_n)$. Therefore, the concept of context is important for defining semantic measures. It is a key element of the semantic model,

based on which the word will be characterized and presented for further comparison.

Syntagmatic context

Syntagmatic contexts concern the study of co-occurrence in sequential word ordering. With this approach, a certain size of word window is typically used to define context, for example, using a window of five nouns, the word "tomato" would be analyzed considering contexts such as "People like vedgetables like tomato because they are a great example of healthy food for them". Considering multi-word windows allows us to not only focus on word combinations, but also to capture words that occur together with larger areas of text and multiple [74, 80] words in between. A variety of alternative linguistic constructs can be used to define syntagmatic context: a sentence, a paragraph, a window of *n* words, or even a window consisting of several characters, to name a few. Different methods can be used to handle context, depending on its definition and setting.

Contexts that correspond to relatively large areas of text, such as documents, paragraphs, or sentences, are usually analyzed by counting the occurrences of each word in each context. This can be used to construct a word-context matrix, e.g. a word-document matrix. Such a semantic model can also be used to compute a word co-occurrence matrix, if necessary, for example, considering that two words that occur in the same context occur simultaneously.

Contexts that are defined by considering shorter text areas, such as those consisting of a window of several words, are not usually used to construct word-context matrices. In this case, the goal is usually to scan documents by moving a predefined window by a certain step, for example, by one word. At each step, the window will be processed considering a central word, which is usually located in the center of the window, such as the word "tomato" in the example above. Given this central word, the matches between the central word and the other words forming the window are analyzed. It is worth noting that intuitively, the size of the window considered will determine the sparsity of the co-occurrence matrix, since the narrower the window, the less two words will tend to co- occur.

A wide variety of approaches and configurations can be used to identify

syntagmatic contexts. Some researchers have proposed to consider directional models based on oriented windows. When estimating co-occurrence, these models also consider the location (left or right) of words that occur together with the main word under study. It has also been proposed to weight adjacent occurrences according to the distance of the words within the window. In some cases, the focal word is not located in the center of the window, and an asymmetric window is used to estimate word co-occurrence.

In general, context window extractors and strategies based on syntagmatic context analysis have low algorithmic complexity. Therefore, they are usually considered the solution of choice for processing large corpora. They usually rely on a few user-defined parameters but have the advantage of being language-independent – except for special preprocessing [4-5] and certain configurations that take into account-oriented windows.

Taking a different approach, syntagmatic relations between words can also be studied by identifying specific lexical patterns. As an example, to study the co-occurrence between two nouns, one can analyze specific types of lexical relations, for example, considering two words w 1 and w 2, one can assess their commonality by analyzing the following patterns:

- Hyponymy, Hypernymy: (w_1, is w_2), (w_1, such as, w_2)
- Synonymy, antonymy: (w_1, or w_2)

These patterns can be used to define a three-dimensional co-occurrence matrix with dimensions (Word1, Word2, Pattern). This approach can be used to define a semantic model, particularly adapted to the development of parametric semantic measures. Indeed, with the help of a semantic designer of such a model, it is possible to compare words, controlling the importance given to each lexical pattern, and therefore to precisely control the semantics of the scores obtained by the measure.

Paradigmatic context

Paradigmatic contexts refer to indirect occurrences, i.e. situations in which two words occur with the same words, but not together. This often applies to words that establish paradigmatic relationships, such as synonymy, hyperonymy, co-hyponymy,

troponymy, antonymy, or meronymy. Typically, paradigmatic relationships are characterized by an analysis of words that are surrounded by the same words. Therefore, such contexts are usually defined in terms of grammatical dependence between words. In other words, the occurrence of words relative to paradigmatic contexts can be used to assess the distributive similarity of words in terms of lexical interchangeability.

Most paradigmatic patterns have the form (w, ***t***, *x*) or (*x*, ***t***, *w*), where **t** is the studied dependency, *x* is a word that is part of the pattern, and *w* is a word that is used to characterize the pattern. As an example, the pattern (*w*, OBJECT, GROW) can be defined to study words that are subjects of the verb "grow". Using such a pattern, it can be distinguished that the words "plant" and "tree" often occur with such a pattern, for example, in the sentence "The water is the basis for which the roots hold when *the plant* ***grows"*** the word "plant" occurs with such a pattern; there are also many examples of sentences in which the scheme "tree grows" occurs. Conversely, the word "*botany"* never or much less frequently occurs with this particular pattern. Thus, this pattern, which refers to a specific paradigmatic context, can be used to characterize that the words ("plant", "tree") together are probably more semantically related than, for example, the words botany and tree.

To analyze words using paradigmatic contexts, a certain window size and direction are usually considered. In more advanced approaches, specific patterns will be used to characterize subtle variations in contexts, such as lexical-syntactic patterns.

In the context of paradigmatic context analysis, the use of more patterns improves the results of semantic measurements. In addition, syntagmatic contexts and simple sliding window approaches can be used to obtain results almost identical to those obtained using syntactic patterns. However, both syntagmatic and paradigmatic contexts are of particular importance for the analysis of word relations. They are both used to define semantic models on which semantic approaches are based.

To calculate paradigmatic similarity, you need to represent each word as a context and calculate the context similarity. Words with high context similarity are likely to have a paradigmatic relationship.

Representing each word as a context

Let's say there is a simple text document that consists of the following sentences:

Table 4 – Example sentence for processing

Ukrainian	*Я відчуваю тривогу завжди.* *Відчуваю, що щастя переповнювало мене.*

It is clear from the text that there are words that occur in a similar context in each sentence. For analysis, some way of representing the context of a given word is needed. Let's define functions for calculating the context fragments for the word "відчуваю":

Table 5 – The context fragments for the word "відчуваю"

Ukrainian	*Right1("відчуваю") = {"тривогу", "щастя", "що", "настрій", ...}* *Left1("відчуваю") = {"я", "він", "вона", ...}*

Such functions allow us to represent the context of a word as a series of word sets.

Context similarity calculation

The calculation of word context similarity is usually done using the vector semantics method. In this method, each word in the text is assigned a vector that characterizes its semantic meaning based on its context − that is, other words that often occur next to it. Then, using various mathematical algorithms, the vectors of two words can be compared and determine how similar their contexts are.

To calculate the measure of paradigmatic relations, it is necessary to collect the set of similarities of relative contexts. For example:

Table 6 - The set of relative contexts similarities

Ukrainian	*Sim("Відчуваю", "Сприймаю") =* *Sim(Left1("Відчуваю"), Left1("Сприймаю ")) +* *Sim(Right1("Відчуваю"), Right1("Сприймаю "))*

There are many methods for determining the degree of similarity between words. The most common methods are listed below.

Cosine similarity — measures the angle between two vectors representing words in terms of the cosine of the angle between the vectors.

For two vectors (the context vectors of two words), the cosine similarity is defined as the cosine of the angle between them. This value is found by taking the dot products of the two vectors and dividing by the product of their lengths (1):

$$\text{similarity} = \sin(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}, \quad (1)$$

where A and B are the context vectors of the words, and ||A|| and ||B|| are their lengths.

Thus, using vector semantics and cosine similarity, it is possible to calculate the similarity of the context of two words and use this value to solve various linguistic tasks, such as finding synonyms, antonyms, contextual translation, and others.

Estimates the similarity of two text vectors by the angle between them. The closer to 1, the more similar the words are.

Example. In Ukrainian words “Кіт and “Кішка” the values are “Кіт” = (1,2,0), “Кішка” = (1,1,1). Cosine similarity is 0.86 means that this is very similar words.

Euclidean distance — measures the distance between two points in an n-dimensional space, where each dimension corresponds to the frequency of a word in the text (2).

$$euclidean(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2} \quad (2)$$

The greater the distance between two vectors, the lower the similarity score, and vice versa. To normalize the Euclidean distance, since it is not in the range from 0 to 1, we can use Euler's constant.

Measures distance between vectors in multidimensional spacious. Then the smaller, the more similar words.

Example. For such words in Ukrainian “Сонце” = (3,4) and “Зірка” = (7,1) the Euclidean distance is 5, which means that these are quite distant words in context.

The Manhattan metric measures the distance between two points in n-dimensional space as the sum of the absolute differences between the corresponding coordinates of the points.

Sum of the moduli of the differences between the coordinates of words. Smaller distance means greater similarity.

Example. For the words "Дім" = (2,3) and "Будинок" = (5,7) the Manhattan distance is 7, which affects the low similarity of the words.

Jaccard Index (Jaccard similarity) (3) measures the similarity between two sets as the ratio of the number of common elements to the number of unique elements in both sets.

$$Jaccrad(doc_1, doc_2) = \frac{doc_1 \cap doc_2}{doc_1 \cup doc_2} \quad (3)$$

Example. For the words "Кіт" → {К, І, Т} and "Кішка" → {К, І, Ш, К, А} the Jaccard Index is 2/5=0.4, which shows the low similarity of the words.

Sorensen Index (Dice similarity) also measures the similarity between two sets, as the ratio of twice the number of common elements to the sum of the numbers of unique elements in each set.

This measure is like Jacquard, but gives more weight to common elements (4):

$$D(A, B) = \frac{2|A \cap B|}{|A| + |B|} \quad (4)$$

Example. For the words "Ріка" → {Р, І, К, А}and "Річка" → {Р, І, Ч, К, А} the Dice similarity is 4/5=0.8, which shows the high similarity of the words.

Jaro-Winkler similarity distance — used to compare two strings, measuring the distance between them in terms of additional or missing characters, as well as character permutations between the two strings.

It considers the number of common characters and the permutations between them. The more similar letters at the beginning of a word, the higher the value.

Example. For the words "Кіт" and "Кітя" Jaro-Winkler similarity distance is 0.9 which means these words are similar and for the words "Кіт" and "Слон" Jaro-Winkler similarity distance is 0.1 which means these words are different.

Levenshtein distance measures the distance between two strings in terms of insertions, deletions, and character substitutions required to convert one string to another.

The number of operations (insertion, deletion, replacement) required to convert

one word into another.

Example. For the words "Місто" and "Миска" required to define 2 operations: replacement "т" to "с", replacement "о" to "а".

Longest common subsequence finds the longest sequence of characters that occurs in two strings and uses its length as a measure of similarity.

The length of the longest common substring of two words. Used to compare the similarity of texts.

Example. For the words "Веселка" and "Веселий" LCS = "Весел" with length = 5, which means that words are similar.

Algorithm for constructing a matrix of paradigmatic relationships

The paradigmatic relations matrix shows the ways in which words can be substituted for each other within a given context. It is an important tool for analyzing semantic relationships between words and helps identify synonyms, antonyms, hyponyms, and hypernyms.

For example, if there is a matrix of paradigmatic relations for the word "car", then we can see that it can be replaced by the words "machine", "auto", "vehicle", etc. This allows us to understand that these words have a close lexical meaning and can be used in different situations.

The algorithm for constructing a matrix of paradigmatic relationships for a text is described below.

1. Determine the set of the words included in the text.
2. For each word, determine the set of its possible meanings (lexical variants) using a dictionary or corpus of texts.
3. For each word, build a list of words that can replace it in the text, that is, its synonyms or antonyms, using a dictionary or corpus of texts.
4. Construct a matrix of size N x M, where N is the number of words in the text, and M is the total number of possible values for all words in the text.
5. For each word in the text, determine its row in the matrix. For each possible meaning of the word, determine its column in the matrix.
6. Fill the matrix with values using the following algorithm:

- If a word in the text has a possible meaning that matches another word in the text, then the corresponding matrix element is 1.
- If a word in the text has a possible meaning that coincides with the synonym or antonym of another word in the text, then the corresponding matrix element is 0.5.
- If a word in the text has a possible meaning that does not coincide with any other word in the text or with its synonyms/antonyms, then the corresponding matrix element is 0.

7. Use the resulting matrix to analyze paradigmatic relationships between words in the text.

To find the multiple lexical meanings of a word, you can use various sources, such as dictionaries, encyclopedias, linguistic reference books, Internet resources, and others.

How to find multiple lexical meanings. One effective way is to use linguistic programs such as WordNet. WordNet is a lexical database that contains semantic and lexical relationships between English words. It allows you to find multiple lexical meanings of a word and their relationship to other words.

There are also other programs and online resources that allow you to find multiple lexical meanings of a word. For example, Linguee, Merriam-Webster, Oxford English Dictionary, Cambridge Dictionary and others.

In general, to find the multiple lexical meanings of a word, it is necessary to use reliable sources that are based on linguistic research and have appropriate methodological approaches to determining the meanings of words.

The paradigmatic matrix displays lexemes that belong to the same semantic paradigm, for example, a group of emotions of fear, joy, etc. These words are not used simultaneously but can replace each other in the same context. The table below shows an example of paradigmatic connections.

Table 7 – Matrix of pragmatic connections

	fear	anxiety	panic	horror	nervousness
fear	1.0	0.82	0.75	0.80	0.68
anxiety	0.82	1.0	0.73	0.77	0.71
panic	0.75	0.73	1.0	0.85	0.66
horror	0.80	0.77	0.85	1.0	0.62
nervousness	0.68	0.71	0.66	0.62	1.0

Algorithm for constructing a matrix of syntagmatic relations

There are several methods for constructing a syntagmatic relations matrix, depending on what information needs to be obtained and what analysis is being performed. The most common is the frequency analysis method, which is based on the fact that words that appear together more often in a sentence or text have a stronger syntagmatic connection. To construct a syntagmatic relations matrix using this method, it is necessary to calculate the frequency of occurrence of each pair of words and indicate this value in the matrix.

To construct a matrix of syntagmatic relations using the frequency analysis method for a given text, you need:

1. Split the text into words, thus obtaining a list of tokens.
2. Filter the token list to exclude prepositions, conjunctions, particles, and other words that do not have much semantic weight in the context of the sentence (stop words).
3. Create an empty NxN matrix, where N is the number of unique words in the token list.
4. Go through the text and for each word, increase the value by 1 in the matrix cells corresponding to the connections between this word and other words in the sentence.
5. Normalize the values in the matrix to get the relative frequency of each connection between words.

That normalize values in the matrix and get relative frequency each

communication between in words, you need divide each value in the matrix on amount values in the row to get amount relative frequencies for each word in context line.

Thus, it is possible to obtain a matrix of syntagmatic relations, where the values in the cells reflect the frequency of occurrence of each relation between words in the text. This matrix can be used for further text analysis, such as text classification or finding keywords.

A syntagmatic matrix shows which words are often combined together in sentences, i.e. have linear (horizontal) connections in the corpus of texts. This can be constructed based on word co-occurrence.

The construction of such a matrix is considered using the example of the sentence: «*Я боюся темряви, не можу спати*»

Step 1: Splitting the text into tokens.

['я', 'боюся, 'темряви', 'не', 'можу', 'спати']

Step 2: Text processing.

['я', боятись, 'темрява', 'не', могти, 'спати']

Steps 3-4: Constructing the matrix.

First, a list of bigrams is built:

{('я', 'боятись'): 2, ('боятись', 'темряви'): 2, ('темряви', 'не'): 2, ('не', 'могти'): 2, ('могти', 'спати'): 2}

Further, based on these bigrams, a matrix of connections between words can be constructed.

Table 8 – Matrix of syntagmatic relations

	я	боюся	темряви	не	можу	спати
я	—	0.9	0.8	0.85	0.6	0.5
боюся	0.9	—	0.95	0.3	0.2	0.1
темряви	0.8	0.95	—	0.25	0.2	0.1
не	0.85	0.3	0.25	—	0.9	0.6
можу	0.6	0.2	0.2	0.9	—	0.75
спати	0.5	0.1	0.1	0.6	0.75	—

Such a matrix allows the system to detect patterns of expressions characteristic of certain emotional states, in particular fear, stress, anxiety, etc. For example, the high co-occurrence of “Я” + “боюся” + “темряви” indicates a characteristic syntactic pattern of expressing fear.

The relations matrix displays the dependencies between words in a text according to their sequence and syntactic rules. It helps to study and analyze the linguistic structure of a text and find out which words are used together more often, how they emphasize each other, and which grammatical forms correspond to certain syntactic roles.

Thanks to the matrix of syntagmatic connections, it is possible to identify topic and stylistic features of the text [24], identify stable word combinations and phraseological units, as well as investigate linguistic patterns and language structure.

Psycholinguistics examines the mechanisms of language perception and production, considering cognitive, emotional, and social factors. In the context of automatic text analysis, it is important to understand how emotions are encoded in language and how they are perceived by a person at the level of individual experience, linguistic expectations, and social context.

One of the key aspects is that the emotional response to a linguistic stimulus arises not only due to its denotative meaning, but also due to associative, stylistic, and grammatical parameters. For example, words with reduced or increased stylistic marking evoke corresponding emotions even in a neutral context.

There is also a phenomenon of asymmetry in the perception of emotions, with negative emotional markers generally perceived faster and more deeply than positive ones. This effect is important to consider when building models of emotional analysis, as stressful states often correlate with the use of words with negative valence.

Psycholinguistic research also emphasizes the importance of context in interpreting emotions, as the same utterance can carry different emotional load depending on syntactic construction, punctuation, text genre, or even the reader's cultural background. In particular, irony, sarcasm, and euphemisms require deep cognitive processing [36] and cannot be correctly interpreted based on lexical items

alone.

In the case of texts created under stress (e.g., diary entries, reports on military events, appeals for psychological help, posts on social networks, etc.), the structure of the language changes, short sentences become more frequent, verbs of emotions are used (to feel, to be afraid, to be afraid, etc.), and the use of personal pronouns increases. These features can be used as diagnostic markers for automatically detecting the author's stressful state.

The emotional coloring of a text is not limited to the lexical level. It can also be manifested at the level of word formation, morphology, and syntax. These grammatical levels are important for building systems for text mining, as they allow for a more accurate interpretation of the emotional content of messages, especially in conditions of polysemy, homonymy, or contextual variability.

The Ukrainian language demonstrates high morphological and syntactic variability, which is an important resource for identifying emotional manifestations in texts. One of the key linguistic levels where emotional load is realized is word formation. Diminutive and affectionate forms, characteristic of the Ukrainian language, have the ability to convey a positive, sentimental [48] or empathetic color. Forms such as *"дівчинка"*, *"серденько"*, *"вогник"* indicate affective softness and emotional closeness. Another means of expressing emotionality are word-formation models with reinforcing suffixes or prefixes that enhance the expressiveness of adjectives or verbs, in particular *"жахнючий"*, *"надщасливий", "передобрячий"*. Such units serve not only as markers of emotional intensity but can also signal stressful or extreme emotional states. Special attention is paid to ironic word-formation constructions, which, through a specific contextual coloring (for example, *"правдолюбець"*, "*геройчище"*) implement emotional reduction or sarcastic interpretation.

At the morphological level, emotions are conveyed through the choice of parts of speech, tense and personal forms, as well as through modality. Adjectives, verbs and adverbs that have an expressive load are the main carriers of emotional meanings, in particular words like *"боязкий"*, *"сумує"*, *"нервово"*. Of particular importance are the forms of the first-person singular, which often indicate the subjective involvement

of the author in the situation (for example, "*Я не витримую"*, "*Мені страшно"*), which is a characteristic feature of an emotional or stressful state. It is also important to identify modal constructions that indicate uncertainty, desire or anxiety ("*я б хотів", "мені здається", "можливо"*). Considering morphological markers in linguistic preprocessing contributes to a more accurate interpretation of emotions, especially in the conditions of the inflectional nature [59-61] of the Ukrainian language with a large number of word forms.

Syntactic means of implementing emotional expression are manifested through sentence structure, word order, sentence types, and stylistic figures. In particular, ellipsis, exclamations, or broken constructions ("*Жах!", "Нічого не лишилось…"*) indicate a high level of emotional tension or stress. Inversion, i.e., a deviation from the neutral word order, such as in the sentence *"Таке не забувається ніколи!"*, enhances emotional expressiveness. The use of interrogative, exclamatory, or conditional constructions serve as markers of surprise, anxiety, or hope. Repetitions and parallel structures create rhythmic tension that additionally conveys the speaker's emotional state (*"Я боюсь. Я дуже боюсь. Я не можу мовчати.")*. Interpreting such syntactic patterns allows the model to consider not only the local semantics of individual words, but also the global context, which is especially important for detecting complex affective signals in texts.

Thus, effective emotional analysis of Ukrainian texts requires a deep integration of word-formation, morphological, and syntactic levels. The combination of these aspects allows to significantly improve the accuracy of emotion classification and stress detection, especially in conditions of insufficient direct explication of the speaker's emotional state.

Table 9 provides examples of grammatical constructions that can serve as markers of the stress or emotional state of the author of the text.

Taking these grammatical indicators into account in automatic analysis models makes it possible to more accurately detect emotions beyond the lexical level, classify the degree of emotional load of a text, and detect indirect signs of stress, even in tone-neutral vocabulary.

Table 9 – Grammatical constructions as markers of emotional state/stress

Grammatical phenomenon	Example	Interpretation
Diminutive and affectionate forms	"Татусю, не йди!"	Sentimentality, anxiety
Intensive adjectives/adverbs	"Жахливий біль", "безмежно сумно"	High emotional intensity
First person singular	"Я не витримую", "Мені страшно"	Personal, authentic stress
Modal constructions	"Хотів би зникнути", "Міг би втекти"	Insecurity, emotional instability
Exclamations, exclamations	"О, ні!", "Боже мій!"	Reaction to shock/trauma
Repetitions (lexical/syntactic)	"Я боюсь. Я дуже боюсь. Я не можу мовчати."	Emotional confusion, affective escalation
Inversion/emphatic order	"Не словами, а болем мовчить вона."	Poeticity, heightened affect
Ellipsis, interrupted sentences	"Зник. Без сліду. Назавжди…"	Stress, shock, disorganized thinking
Interrogative-rhetorical constructions	"Невже це кінець?", "Чому саме я?"	Internal conflict, emotional turbulence

1.3. Overview of approaches to automatic detection of emotions and stress in text

Automatic detection of emotions and stress in texts is based on three main groups of approaches: lexical-grammatical rules and dictionaries, statistical methods, and machine and deep learning methods.

Lexical and rule-based approaches rely on pre-compiled dictionaries (lexicons) that contain a list of words with a corresponding emotional valence (positive, negative, neutral) or emotional class (joy, fear, anger, etc.). Examples of such lexicons are NRC Emotion Lexicon, SentiWords, AFINN, and for the Ukrainian language – UkrSentimentLexicon.

Models of this type of work by counting the number of words with corresponding emotional meanings, then summing or balancing the tones. The rules can consider contexts of negation (e.g., “не радий” ≠ “радий”), inversion, or intensity using modifiers (" дуже щасливий" > "щасливий").

The advantages of this approach are its interpretability, speed of implementation, and low data requirements. The disadvantages are limited flexibility, lack of learning ability, and low efficiency in cases of irony, slang, and implicit expression of emotions.

Statistical approaches are based on the use of computational features of the text - such as word frequency, specific grammatical constructions, density indices, sentence complexity. These models use algorithms such as TF-IDF, Latent Dirichlet Allocation, clustering or factor analysis [78].

Topic modeling, such as LDA, can identify themes associated with fear, anger, or tension. Frequency analysis of emotionally charged tokens can also identify areas of text with increased affectivity.

Statistical models are easy to implement and effective when there is a limited amount of marked-up data. However, they do not consider deep semantics and syntax, which limits their accuracy in complex texts.

Modern emotional analysis systems mainly use machine learning methods, especially deep learning. Traditional algorithms, such as Support Vector Machine, Naive Bayes, XGBoost, etc., work on manually generated features [66] (e.g., word frequencies, grammatical patterns), while neural networks can automatically learn relevant features from text.

Recurrent neural networks (LSTM, GRU) that process word sequences and take order into account are particularly effective. An even new generation of models — transformers, such as BERT, RoBERTa, XLM-RoBERTa — implement attention [67] mechanisms that allow the model to focus on the most informative parts of the text.

In the context of the Ukrainian language, multilingual models [37] XLM-R, mBERT, and specialized local ones, such as UkrBERT, are already available, which are successfully used to solve the problems of tonal analysis, emotion classification, and stress indicator detection [38].

Despite their high accuracy and flexibility, deep learning methods have limited interpretability and require large training corpuses.

1.4. Features of Ukrainian language processing in NLP

Natural language processing in the context of Ukrainian faces several linguistic, technical, and resource challenges that significantly distinguish it from processing English or other languages with a high level of digital support. This section examines the key features of the Ukrainian language that affect the construction of systems for automatic text analysis, in particular in the tasks of emotional classification, stress marker extraction, and topic modeling.

The Ukrainian language is characterized by a developed system of declension and conjugation, which leads to a large number of word forms. In particular, nouns change in seven cases, and verbs in tenses, persons, genders and species. This requires the use of accurate morphological analysis and lemmatization, which is critically important for creating reliable language models. For example, the forms "говорив", "говорила", "говорили" have the same lemma, but differ in grammatical categories, which is important to consider in semantic analysis.

One of the defining syntactic features of the Ukrainian language is the free word order in a sentence. This means that the syntactic roles of subject, predicate, and object can be determined not only by position, but also by morphological features. For example, the sentences "Марія побачила Івана" and "Івана побачила Марія" have the same grammatical components but differ in the focus of information. Such flexibility complicates the tasks of syntactic parsing, in particular when constructing dependency trees.

Although more Ukrainian language tools have appeared in recent years (e.g. Stanza, UD-Pipe, lang-uk, Ukrainian BER), the Ukrainian NLP community still faces a lack of large, high-quality annotated corpora, specialized lexicons of emotional words and stress markers. This limits the possibilities of training powerful models and overall progress in high-level tasks such as emotional interpretation, dialog systems or intention recognition.

The Ukrainian language has a high productivity of word-forming processes. Suffixes, prefixes and other word-forming elements form a large number of lexemes,

which can differ in emotional coloring. For example, the words "жалість", "жалібність", "жалюгідність" have a common semantic basis, but a different emotional spectrum. This creates a need for deep semantic modeling and construction of paradigmatic series based on vector representations of words.

In informal communication, the Ukrainian language often uses orthographic simplifications, dialecticism, jargonism, and transliteration, which complicates automatic analysis. For example, "шо" instead of "що" or "взагалі-то" in a colloquial style may have an intonation load that indicates an emotional context. Processing such cases requires contextual consideration and adaptation of the pre-processing stages.

For the Ukrainian language, models that consider morphological structure have proven effective, in particular FastText, which works at the symbol and subword level. Multilingual transformers, such as mBERT, XLM-R, demonstrate satisfactory results, but are trained in many languages. Ukrainian-language transformers provide significantly better performance in specific tasks. SlavicBERT, UBERT, trained on local news corpora, Wikipedia, and social media posts.

Analysis of the specificity of the Ukrainian language from the NLP perspective shows that linguistic features, such as morphological richness, syntactic flexibility, and word formation variability, form a unique linguistic environment that requires adapted approaches to processing. Despite existing limitations, the development of Ukrainian-language resources and models opens new opportunities for studying such phenomena as emotionality and stress in the text, which are relevant both in applied and psycholinguistic aspects.

Conclusions

Effective intellectual analysis of Ukrainian-language texts for the detection of emotions and stress requires a comprehensive approach. It is based on an understanding of the complex psychophysiological processes of emotions, their linguistic markers at the lexical, morphological, syntactic, and punctuation levels of language, as well as on the differences between emotion and stress. Critically important is a deep linguistic analysis that includes both paradigmatic and syntagmatic connections between words that form the meaning and grammatical coherence of sentences.

The concept of context is fundamental to fixing the meaning of a word, and its definition can vary from a document to a small window of words, which affects the methods of calculating similarity. The use of vector semantics methods and various similarity metrics allows us to quantitatively assess linguistic relations and build matrices of paradigmatic and syntagmatic relations.

Psycholinguistic research emphasizes the importance of contextual interpretation of emotions, as well as language features characteristic of stressful states, such as changes in sentence structure, the use of first-person pronouns, and specific grammatical constructions. This is especially relevant for the Ukrainian language due to its rich morphology, free word order, high word-formation productivity, and features of informal communication. Despite the lack of large, qualitatively annotated corpora, the development of Ukrainian-language transformers and adapted processing approaches, considering these linguistic features, opens significant opportunities for progress in this field.

Automatic detection of emotions and stress is implemented using lexical-grammatical rules and dictionaries, statistical methods and, especially effectively, machine and deep learning methods. The latter allows them to automatically learn relevant features from the text, despite their "black box" nature and high data volume requirements. Integration of word-formation, morphological and syntactic levels is key for more accurate interpretation of emotions and detection of stress in texts in Ukrainian, especially when the emotional state is expressed indirectly.

KAPITEL 2 / *CHAPTER 2*
DATA PREPARATION FOR ANALYSIS OF UKRAINIAN TEXTS

2.1. Texts preprocessing

Text preprocessing is a key component of many text mining algorithms. For example, the traditional approach to text categorization includes preprocessing, feature extraction, feature selection, and classification stages. The main text preprocessing technologies are listed below.

Tokenization. Tokenization is the first step in text mining. It is the process of breaking strings into tokens, which in turn are small structural units (tokens). Tokenization involves three stages that break a complex sentence into words, determine the importance of each word in relation to the sentence, and provide a structural description of the input sentence. As a result of tokenization, a list of tokens can be generated, which is used for further processing.

One approach to tokenization is to break this sentence into words and punctuation marks (i.e. tokens). Identifying words in a language that uses the Latin or Cyrillic alphabet is relatively easy, since spaces are used. However, punctuation marks can be a bit more ambiguous. For example, a period can indicate the end of a sentence or an abbreviation, etc. Other languages, such as Mandarin, do not use spaces to indicate separation between words. They require a different approach to determining what constitutes a word. Finally, languages like German deal with verbs in a unique way. For example, if a word has a separable prefix (such as “викинути”), then German grammar dictates that it be moved to the end of the sentence. Thus, “я викину геть сміття” becomes literally “я сміття геть-викину.”.

Tokenization example is displayed below on Figure 1.

Садок вишневий коло хати, хрущі над вишнями гудуть
↓

садок	вишневий	коло	хати	хрущі	над	вишнями	гудуть

Figure 1 - Example of tokenizing words in a sentence

Finding the frequency of words occurrence in the text. Finding the frequency of the occurrence of words in a text is essentially a simple count of the occurrences of each specific word/token in the text.

Example of finding the frequency of words occurrence in a text is shown below (Figure 2).

Садок вишневий коло хати, хрущі над вишнями гудуть
↓

садок - 1	вишневий -1	коло - 1	хати - 1	хрущі- 1	над- 1	вишнями- 1	гудуть- 1

Figure 2 – Example of counting the frequency of occurrence of words in a sentence

Removing stop words. Stop words are words that occur particularly frequently in a text corpus and are thus considered to be rather uninformative (e.g. words like, “як”, “і”, “або”, ...). One approach to removing stop words is to search a language dictionary of stop words. An alternative approach is to create a stop list by sorting all words in the entire text corpus by frequency. The stop list after being transformed into a set of unnecessary words is then used to remove all those words from the input documents that are among the first n words in this stop list.

Example of removing stop words is shown below (Figure 3).

Садок вишневий коло хати, хрущі над вишнями гудуть
↓

садок	вишневий	хати	хрущі	вишнями	гудуть

Figure 3 – Example of removing stop-words

Stemming. Stemming is the process of reducing compound (or sometimes derivative) words to their base words. or root form - usually the written form of a word. The stem (root) does not necessarily have to be identical to the morphological root of the word; it is usually sufficient that related words appear in the same stem, even if the stem itself is not a valid root.

There are several types of stemming algorithms that differ productivity and accuracy of work.

A simple stemmer looks up the inflected form in a lookup table. The advantages of this approach are that it is simple, fast, and handles exceptions easily. The disadvantages are that all compound forms must be listed explicitly in the table: new or unfamiliar words are not processed, even if they are perfectly regular, and the table can be large. For languages with simple morphology (English), the size of the tables is moderate, but highly inflected languages such as Turkish may have hundreds of potential inflected forms for each root.

Stemming example is demonstrated below (Figure 4).

Садок вишневий коло хати, хрущі над вишнями гудуть							
↓							
сад	вишнев	кол	хат	хрущ	над	вишн	гудут

Figure 4 – Example of using the stemming method

Lemmatization. This is a more complex approach, based on determining the root of a word by lemmatization. The first step of this algorithm is to determine the parts of speech in a sentence, the so-called POS tagging. In the second step, stemming rules are applied to the word according to the part of speech [53]. That is, the words “пальне” and “вітальне” should pass through different chains of rules, because “пальне” is a noun, and “вітальне” is an adjective. Theoretically, stemming algorithms based on lemmatization should have very high quality and a minimal percentage of errors, but they are very dependent on the correct recognition of parts of speech.

In simple terms, it is the process of converting a word into its base form. The difference between stemming and lemmatization is that lemmatization takes the context into account and converts the word into a meaningful base form, while stemming simply removes the last few characters, which often leads to incorrect meanings and spelling errors.

For example, lemmatization would correctly define the base form “турбуватись” to “турбота”, whereas stemming would cut off the part and turn it into “турбу”.

Example of lemmatization is shown below (Figure 5).

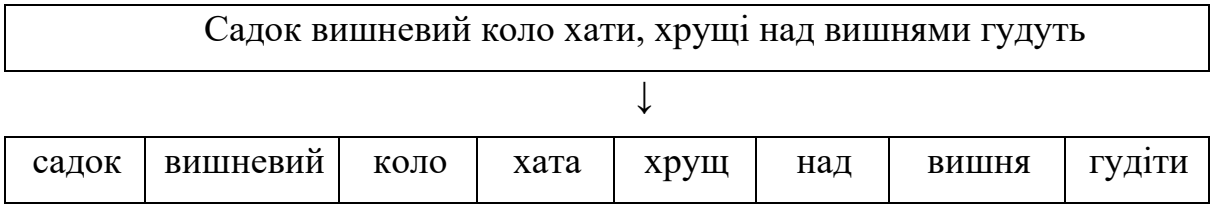


Figure 5 – Example of using the lemmatization method

The difference between stemming and lemmatization is given in the table 10.

Table 10 – The difference between stemming and lemmatization

Слово	**Стемінг**	**Лематизація**
садок	сад	садок
вишневий	вишнев	вишневий
коло	кол	коло
хати	хат	хата
хрущі	хрущ	хрущ
над	над	над
вишнями	вишн	вишня
гудуть	гудут	гудіти
плугатарі	плугатар	плугатар
з	з	з
плугами	плуг	плуг
йдуть	йдут	йти
співають	співа	співати
ідучи	ідуч	ідучи
дівчата	дівча	дівчина
а	а	а
матері	матер	мати
вечерять	вечерят	вечеряти
ждуть	ждут	ждати

Parts of speech (POS). In corpus linguistics, part-of-speech tagging (POS tagging or PoS tagging or POST tagging), also called grammatical tagging, is the process of marking a word in a text (corpus) as belonging to a particular part of speech, based on both its definition and context.

Originally done manually, POS annotation is carried out in the context of computational linguistics, using algorithms that associate discrete terms as well as

hidden parts of speech using a set of descriptive tags. POS annotation algorithms can be divided into two distinct groups: rule-based and stochastic.

The process of part-of-speech labeling is more complicated than simply creating a list of words and their parts of speech, because some words can represent more than one part of speech at different times, and because some parts of speech are complex or unpronounced. This is not uncommon — in natural languages (unlike many artificial languages), a large percentage of word forms are ambiguous.

Each language has its own set of parts of speech. However, there are obviously many more categories and subcategories. For nouns, plural, singular, and possessive forms can be distinguished. In many languages, words are also marked according to their "cases" (role as subject, object, etc.), gender, and so on; while verbs have markings for tenses, types, and other things. Linguists distinguish parts of speech from varying degrees of precision, reflecting the chosen “marking system”.

For English, there are between 50 and 150 separate part-of-speech notations for English. For example, NN for singular common nouns, NNS for plural common nouns, NP for singular proper nouns (see the part-of-speech notations used in Brown's collection Corpus). Work on stochastic methods for Koine markup used over 1000 part-of-speech notations and found that about the same number of words were polysemous as in English. The morphosyntactic descriptor in the case of morphologically rich languages is usually expressed using a very short mnemonic, e.g. Ncmsan Part of speech = noun, Type = common, Gender = male, Number = single, Case = accusative, Entity = non-living (no).

POS tagging example is demonstrated below (Figure 6).

Садок вишневий коло хати, хрущі над вишнями гудуть
↓

садок noun:inanim:m:v_naz	вишневий adj:m:v_naz	коло prep	хати noun:inanim:p:v_zna	хрущі noun:anim:s:v_naz	над prep	вишнями noun:inanim:s:v_oru	гудуть verb:imperf:pres:s:3

Figure 6 – Example of using the POS-tagging method

Chunking. Chunking means collecting individual pieces of information and grouping them into larger pieces. In the context of NLP and text mining, fragmentation means grouping words or tokens into new structural associations. Chunking works based on POS tags. It uses POS tags as incoming data and provides formation fragments as weekend data. In other words, fragmentation means grouping words/tokens into pieces.

Chunking can break sentences into phrases that are more useful than individual words and produce meaningful results. Fragmentation is very important if there is a need to extract information from the text, such as locations, people's names. (essence extraction)

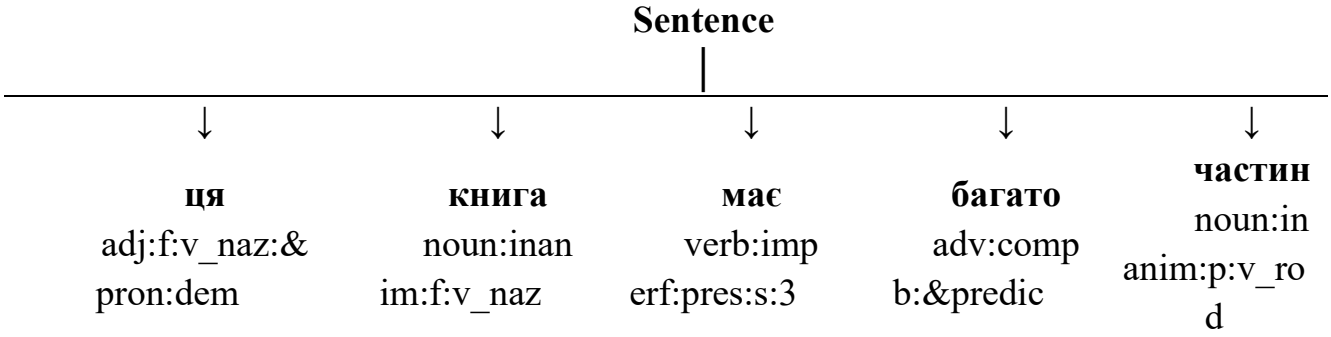
A sentence typically has a hierarchical structure consisting of the following components.

sentences → clauses → phrases → words

A group of words makes up a phrase, and there are five main categories.

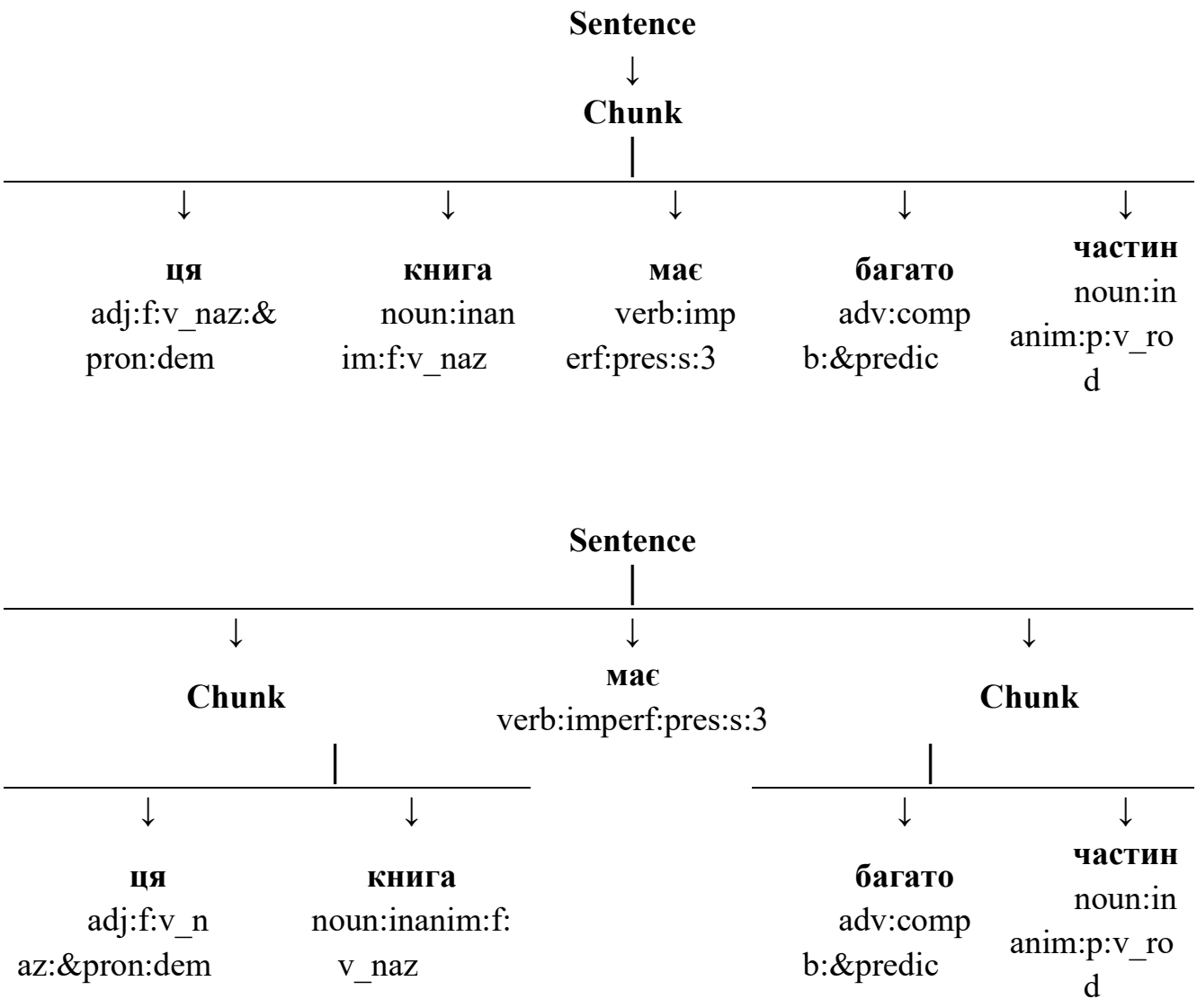
Below is an example of chunking flow.

1) Converting a sentence into a flat tree.



2) Creating fragments using this tree.
3) Creating a RegexpChunkParser by parsing the grammar using RegexpParser.
4) Applying the created chunking rule to a ChunkString, which breaks the sentence into chunks.
5) Splitting a large fragment into smaller ones using defined fragmentation rules.
6) ChunkString is converted back into a tree using two subtrees.

After completing some or all (depending on the tasks set) stages of text preparation, you can proceed to the next stages of text analysis.



2.2. Data Annotation for Emotional Analysis

Text data annotation is one of the key stages in the process of creating intelligent analysis systems, in particular for detecting the emotional coloring of texts and diagnosing stress states. It involves assigning certain emotional labels to individual texts or their fragments that correspond to certain categories (emotions, tone, stress level, etc.). Qualitative data annotation ensures the reliability of classification models, topic modeling, tone analysis, and vector generalization of semantic relationships.

Manual annotation is performed by expert annotators who interpret the emotional context of texts according to pre-agreed instructions and a classification scheme. It is the most reliable, albeit resource-intensive, method of data markup.

This approach is characterized by high quality and contextuality, as humans can detect subtle emotional nuances, recognize irony, sarcasm, allusions, and complex syntactic constructions. Manual annotation is also flexible in interpreting ambiguous formulations.

However, a significant disadvantage is the high cost and time-consuming process of manual data annotation. To ensure data consistency, detailed instructions are required, which is an additional task. With large amounts of data, manual annotation has low scalability.

To assess the reliability of the markup, it is advisable to use the consistency metrics Cohen's Kappa coefficient or Fleiss' Kappa coefficient. They help assess how much the subjective decisions of different experts agree — and whether this agreement is higher than expected by chance.

Cohen's Kappa coefficient is used when there are two annotators who classify objects into one of the fixed categories (e.g. “stressful”, “neutral”). The problem is that the usual accuracy does not consider chance agreement. For example, if 80% of the texts are “neutral”, even chance agreement may be high, which is what this coefficient considers. Below is the formula for calculating Cohen's Kappa coefficient (5).

$$k = \frac{p_0 - p_e}{1 - p_e}, \quad (5)$$

where, p_0 — actual proportion of approvals, p_e — the expected proportion of matches if annotators are chosen randomly.

Table 11 provides an interpretation of Cohen's Kappa coefficient values.

Table 11 – The interpretation of Cohen's Kappa coefficient values

Coefficient value	Consistency
< 0.00	worse than random
0.01–0.20	very weak
0.21–0.40	weak
0.41–0.60	moderate
0.61–0.80	significant
0.81–1.00	almost full (very high)

Fleiss Kappa coefficient is used when there are more than two annotators (e.g., 3–5 people rate each text). This coefficient generalizes the idea of Cohen's Kappa in the case of *n* annotations per item. The interpretation of the Fleiss Kappa coefficient values is similar to Cohen's Kappa: from 0 as random agreement to 1 as perfect agreement.

The use of the above coefficients will serve to confirm the reliability of the corpus, if the coefficient value is low, then you need to review the instructions or data. Also, only with a high value *k*can you train accurate models. Regarding the aspect of the task itself, a low coefficient value may indicate its ambiguity and, accordingly, the subjectivity of the task.

In studies on emotional analysis and stress state marking in Ukrainian texts, the use of specialized tools for data annotation plays an important role. In particular, Doccano, brat, Label Studio and Prodigy are the most popular platforms for manual or semi-automatic text markup.

Doccano is a web interface with multi-user markup support that allows you to create projects for classification, sequential markup (NER), and sentiment analysis. B rat focuses on deep syntactic and semantic markup with dependency visualization. Label Studio provides high flexibility − it can be adapted to different types of tasks: from classification to image and sound analysis, which is especially useful for multimodal data. Prodigy is a commercial tool that allows interactive data annotation with active learning support, where the model “prompts” the annotator with the least certain examples.

Such tools ensure high quality markup, control of consistency between annotators, and scalability of the corpus, which is critical for building reliable emotional analysis models.

Automatic annotation involves the use of lexical, statistical, or machine learning methods to assign emotional labels to texts without human intervention at each stage. Automatic annotation is performed using lexical methods, transformer models, or by applying classical machine learning methods.

This approach is characterized by high processing speed for large corpora. However, automatic annotation depends on the quality of the initial corpus and

demonstrates low accuracy in cases of emotional ambiguity, contextual ambiguity or stylistic figures (sarcasm, hyperbole). Therefore, it is advisable to carry out further manual verification of the results and adapt the models to the specifics of the Ukrainian language by further training on local data.

To detect stress markers in texts, it is necessary to have a specialized corpus divided into at least two categories like stressful and neutral messages. This allows the model to be trained to differentiate texts with signs of psychological stress.

Sources for stressful texts can include psychological assistance and support platforms, social networks and forums on the topic of psycho-emotional support, anonymous blogs, complaints, appeals, posts showing signs of emotional tension, personally collected examples with verbal indicators of stress. Sources for neutral texts can include information resources such as news and encyclopedias, technical documentation, instructions, legislative acts, descriptive or scientific texts without emotional load.

The annotated corpus for the system for determining the emotional coloring of texts and determining stress can be stored according to the structure presented in table 12.

Table 12 – The structure of the annotated corpus

Element	Description
Text	Message, fragment or complete document
Emotional label	stress / neutral / undefined
Emotion type	anxiety, anger, sadness, fear, etc. (as needed)
Tonality	positive / negative / neutral
Metadata	source, date, topic, anonymity

Describing the structure of an annotated corpus for emotional analysis of Ukrainian texts requires a clear specification of each markup element, which ensures both linguistic relevance and technical suitability of the corpus for further processing, analysis, or machine learning.

The main unit of analysis in the corpus is a text fragment, which can be a separate sentence, paragraph, user message or a full-fledged document. Each text is presented

in its original form without censorship of emotional coloring. The authenticity of the linguistic material and the use of the Ukrainian language as the main one are considered, with the fixation of possible inclusions of other languages or elements of slang, dialects, etc.

The emotional label captures the basic classification of the text regarding the presence of emotional tension or psycho-emotional discomfort. It is a key parameter for two-class or three-class classification tasks. The following categories are provided:

- "stress" − the text contains linguistic markers of a stressful state;
- "neutral" − no signs of emotional imbalance;
- "undefined" − cases of ambiguous interpretation or disagreement between annotators.

The emotion type parameter is used to refine the emotional label by categorizing basic affective states. Annotation of the emotion type is useful for building multi-class classifiers and psycholinguistic analysis. Typical categories include anxiety, fear, anger, sadness, despair, guilt, embarrassment, etc. In some cases, mixed or complex emotions can be specified.

Tone determines the overall polarity of the text—positive, negative, or neutral, regardless of the specific emotion type. This parameter allows to separate, for example, emotionally positive experiences, like "I'm excited", from negative, as "I'm scared", and neutral messages, like "I'm reading the news".

Metainformation serves as a basis for analyzing the sources, sociocultural context, and ethical relevance of data. Metadata includes:

- source of the message (forum, social network, messenger, etc.);
- date of creation or publication of the text;
- topic field (e.g., health, war, work, relationships);
- the degree of anonymity, which determines whether the text was subject to de-identification.

The clear structure of the annotated corpus with multi-level markup allows creating a high-quality training sample for emotional analysis tasks, in particular, automatic detection of stress states. This approach helps to increase the reliability of

the results, facilitates statistical verification, and ensures compatibility with modern natural language processing models.

Effective annotation of text data is the basis for creating high-quality tools for emotional analysis of Ukrainian-language texts. The combination of manual and automated methods allows optimizing the process of creating large corpora with high semantic accuracy. Particular attention should be paid to the selection of sources for stressful messages, ensuring ethical processing of private texts, as well as formalizing the criteria for assigning them to a particular emotional category.

2.3. Linguistic resources for the Ukrainian language

Linguistic resources for the Ukrainian language are fundamental components for the development of modern technologies aimed at processing, analyzing, generating, and learning language. These resources are the basis for progress in the field of natural language processing, linguistic research, the creation of educational tools, and many other applied solutions.

Linguistic resources can be grouped into several main categories, each of which plays a unique role.

The first category is corpora. Corpora are large, structured collections of linguistic data, which can be written or spoken. They are collected from a variety of sources, such as fiction and non-fiction, news, web pages, and recordings of spoken language.

There are general-purpose corpora that cover a wide range of topics and styles, such as the Ukrainian National Corpus of Texts. Alongside them, there are specialized corpora that focus on specific topics, such as medical, legal, or specific genres, such as journalism or social media.

Of particular value are annotated corpora that contain additional linguistic information. This can be morphological markup, like indicating part of speech, case, gender; syntactic markup, as representing syntactic relations between words; semantic annotation, like identifying named entities, word meanings or pragmatic markup (determining emotions or intentions). Parallel corpora containing texts and their

translations into other languages are indispensable for the development of machine translation systems and comparative linguistic research.

The next category is lexical resources, which include organized databases that provide detailed information about words and their properties. These include various dictionaries. Explanatory dictionaries, such as the Ukrainian Dictionary in 11 Volumes (SUM-11), explain the meaning of words, while translation dictionaries provide equivalents in other languages. Specialized dictionaries cover the terminology of specific industries, as well as groups of synonyms, antonyms, and phraseological units.

In addition to dictionaries, important lexical resources are thesauri (such as resources similar to WordNet for the Ukrainian language), which hierarchically and associatively link words by their meanings. Ontologies and taxonomies create formalized knowledge models, describing concepts and their relationships. There are also lexical semantics databases, which detail the meanings of words and their semantic roles. Lists of stop words, i.e. frequent words that are usually excluded from processing due to their low semantic value, are also helpful.

Grammatical resources can also be distinguished as a separate category. Grammatical resources describe the structural rules of the Ukrainian language at different levels. They include formal grammars that define the syntactic structure of sentences. Dependency trees that visualize the syntactic structure of sentences are also important components. In addition, there are rules of morphological analysis and generation that regulate the changes of words by case, gender, number, etc.

Another category of linguistic resources are software tools and libraries that provide functionality for automated work with linguistic data. These can be morphological analyzers and tokenizers (for example, pymorphy2 for Ukrainian), which break text into words and determine their grammatical features. Parsers build syntax trees of sentences, and named entity recognizers (NER) identify and classify important objects in the text.

This category also includes complex systems such as machine translation systems that automatically translate texts, text-to-speech systems that convert text into speech, and speech-to-text systems that convert speech into text. In general, various NLP

libraries (e.g. SpaCy, NLTK, Transformers) simplify the development of natural language processing applications, although their support for the Ukrainian language may vary.

The machine learning dataset category contains specialized datasets designed to train machine learning algorithms to solve specific NLP tasks. Examples of such datasets are datasets for text classification, for question answering systems, for text summarization, or for text generation.

Linguistic resources for the Ukrainian language play a key role in many areas. They are a driving force for the development of NLP technologies, allowing the creation of more effective machine translation systems, chatbots, virtual assistants, and text analysis tools.

In the field of linguistic research, these resources allow for in-depth analysis of language, identifying patterns, studying its evolution and variations. In education, they contribute to the development of interactive teaching materials, spelling and grammar checking systems, and language learning programs. In addition, linguistic resources help in cultural preservation by popularizing the Ukrainian language and its unique features. They also improve the accessibility of information, for example, through speech synthesis systems for people with disabilities.

Active development and support of high-quality linguistic resources is critically important for the further development of the digital infrastructure of the Ukrainian language and its integration into the global technological space.

Conclusions

The development and maintenance of high-quality linguistic resources is a fundamental prerequisite for deepening the digital infrastructure of the Ukrainian language and its integration into the global technological space. Text preprocessing is one of the central stages of text mining algorithms. This process includes a sequence of operations aimed at preparing raw text material for further analysis.

Text data annotation is an integral stage in the development of intellectual analysis systems, in particular for determining the emotional coloring of texts and diagnosing stress states. Manual annotation provides the highest reliability due to the ability of annotators to recognize subtle emotional nuances, irony, sarcasm and complex syntactic constructions. However, this method is resource-intensive, expensive and limited in scalability for large volumes of data. Automatic annotation offers high processing speed for large corpora. However, its accuracy depends significantly on the quality of the initial corpus and is low in cases of emotional ambiguity or stylistic figures. In this regard, further manual verification of the results and adaptation of models is advisable.

To detect stress markers, it is critical to have a specialized annotated corpus, divided into categories of “stressful” and “neutral” messages. The structure of such a corpus should clearly specify each markup element, including text, emotional label, emotion type (if applicable), tone (positive/negative/neutral), and metadata.

Optimization of the process of creating large corpora with high semantic accuracy is achieved by combining manual and automated annotation methods. This is the basis for the development of high-quality tools for emotional analysis of Ukrainian-language texts.

KAPITEL 3 / *CHAPTER 3*
ANALYSIS OF EMOTIONAL MARKS AND DETERMINATION OF STRESS

3.1. Methods for identifying emotional markers

Common classifiers and learning algorithms cannot directly process text documents in their original form. Therefore, at the preprocessing stage, documents are transformed into a more manageable form. Typically, documents are represented by vector elements. A feature is simply an entity without an internal structure − a dimension in the feature space. A document in this space is represented as a vector − a sequence of features and their weights.

The most common "bag of words" model simply uses all the words in a document as features, and therefore the size of the feature space is equal to the number of distinct words in all documents. Methods for weighting features can vary. The simplest is a binary file, in which the weight of an item is either one if the corresponding word is present in the document, or zero otherwise. More complex weighting schemes are possible that considering the frequency of words in the document, the category, and the entire collection. The most common scheme is the TF-IDF scheme.

Feature extraction

In NLP, words can be represented in different ways, depending on the purpose of text processing. Below is a description of several basic approaches to extracting features from text.

One-Hot Encoding. A method in which each word is represented as a vector, where only one coordinate has the value 1 and all others have the value 0.

Example 1. Given a dictionary {"happiness", "sadness", "joy"}, the corresponding vector is given below.

"happiness" → [1, 0, 0]

"sadness" → [0, 1, 0]

"joy" → [0, 0, 1]

The disadvantage of this method is the large vectors with a large dictionary.

Example 2. Below is a text in Ukrainian and its vector (Figures 7-8).

```
textsUA = [
    "Я відчуваю тривогу завжди.",
    "Відчуваю, що щастя переповнювало мене."
]
```

Figure 7 – The example of a text in Ukrainian

```
One-Hot Encoding (UA):
   відчуваю  завжди  мене  переповнювало  тривогу  щастя  що
0         1       1     0              0       1       0   0
1         1       0     1              1       0       1   1
```

Figure 8 – The One-Hot Encoding vector of the text in Ukrainian

Bag of Words. A method in which text is represented as a frequency vector of words. The disadvantage is that bag of words ignores the order of words in texts.

Example 1. Text 1: "Кіт спить на дивані", Text 2: "Собака грає у дворі". Table 13 demonstrates bag-of-words matrix for both texts.

Table 13 – The bag-of-words matrix

Word	Text 1	Text 2
Кіт	1	0
спить	1	0
на	1	0
дивані	1	0
Собака	0	1
грає	0	1
у	0	1
дворі	0	1

Example 2. For the text in the figure below, the corresponding bag of words representation is given (Figures 9-10).

```
textsUA = [
    "Я відчуваю тривогу завжди.",
    "Відчуваю, що щастя переповнювало мене."
]
```

Figure 9 - The example of a text in Ukrainian

```
Bag of Words (UA):
   відчуваю  завжди  мене  переповнювало  тривогу  щастя  що
0         1       1     0              0       1       0   0
1         1       0     1              1       0       1   1
```

Figure 10 - The BoW vector of the text in Ukrainian

TF-IDF (Term Frequency - Inverse Document Frequency). Estimates the importance of words in documents, considering their frequency. TF (term frequency — word frequency) the ratio of the number of occurrences of the selected word to the total number of words in the document. Thus, the importance of the word t_i is estimated within the selected document.

$$TF = \frac{n_i}{\sum_k n_k}, \quad (6)$$

where n_i is the number of occurrences of the word in the document, and in the denominator is the total number of words in the document.

IDF (inverse document frequency — inverse document frequency) the inverse of the frequency with which a word occurs in the documents of a collection. Using IDF reduces the weight of commonly used words.

$$IDF = \log \frac{|D|}{|(d_i \supset t_i)|}, \quad (7)$$

where |D| — number of documents in the collection; $|d_i \supset t_i|$ — the number of documents in which the word t_i occurs (when $n_i \neq 0$).

The choice of logarithm base in the formula does not matter, because changing the base will change the weight of each word by a constant factor, i.e. the weight ratio will remain unchanged.

In other words, the TF-IDF indicator is the product of two factorsTF and IDF.

Words with a high frequency of occurrence within a document and a low frequency of use in other documents in the collection will receive a higher TF-IDF weight.

This method is useful for information retrieval.

Example. For the text in the figure below, the corresponding TF-IDF representation is given (Figures 11-12).

```
textsUA = [
    "Я відчуваю тривогу завжди.",
    "Відчуваю, що щастя переповнювало мене."
]
```

Figure 11 - The example of a text in Ukrainian

```
TF-IDF (UA):
   відчуваю    завжди      мене  переповнювало   тривогу     щастя        що
0  0.449436  0.631667  0.000000       0.000000  0.631667  0.000000  0.000000
1  0.335176  0.000000  0.471078       0.471078  0.000000  0.471078  0.471078
```

Figure 12 - The TF-IDF vector of the text in Ukrainian

Word Embeddings (Word2Vec, GloVe, FastText). Word embeddings are methods for representing words as dense numerical vectors that capture semantic similarity between lexical items. Unlike surface models, embeddings allow the model to understand the relationship between words based on their context in the corpus. This is especially important for sentiment analysis, where the meaning of a word is closely related to the environment in which it is used.

Word2Vec is one of the first neural models that allows you to generate vectors of words based on their context. It was proposed by the Google team, it has two architectures: CBOW (Continuous Bag of Words), which predicts a word based on context, and Skip-gram, which, on the contrary, predicts the context for a given word. The advantages of Word2Vec are high semantic accuracy and efficiency in large corpora. However, this model does not consider the morphological structure of the word which is especially critical for the Ukrainian language and has one vector representation per word without considering polysemy.

Example.Below is a vector representation for the word "відчуваю" (Figure 13).

```
Word2Vec (UA) - Вектор слова 'відчуваю':
[ 0.05455794  0.08345953 -0.01453741 -0.09208143  0.04370552  0.00571785
  0.07441908 -0.00813283 -0.02638414 -0.08753009]
```

Figure 13 - Word2Vec vector representation for the word "відчуваю"

FastText is a model developed by Facebook AI Research, which works not only with words, but also with symbolic n-grams. This allows you to form vectors even for unfamiliar words (OOV – out of vocabulary), which is critically important for inflectional languages, particularly Ukrainian. This model is characterized by the fact that the word is divided into symbolic subsequences (for example, "щастя" → <ща, щас, аст, стя> etc.). The representation of a word is the average vector of its n-grams.

This model considers morphology, works effectively with derived forms, diminutives, dialectisms, and is characterized by resistance to lexical variability, which is important for recognizing emotional suffixes, for example, “серденько”, “страшнючий”. However, FastText requires more resources in training and may have a larger model size.

Example. Below is a vector representation for the word “відчувати”(Figure 14).

```
FastText (UA) - Вектор слова 'відчуваю':
[-0.00518999  0.00641658  0.00978472 -0.02427484  0.01583607  0.00968476
  0.00215367 -0.01582324  0.00458109 -0.01402789]
```

Figure 14 - FastText vector representation for the word “відчуваю”

GloVe (Global Vectors for Word Representation), a model developed at Stanford, combines the Word2Vec approach with co-occurrence statistics, i.e. it uses a co-occurrence matrix to compute vectors that preserve global semantic patterns. GloVe preserves both local and global context, which contributes to better quality vectors for infrequently used words. Despite its significant advantages, this model does not support morphological segmentation and, like Word2Vec, is designed for “static” vectors.

Transformer models (BERT, GPT). In modern natural language processing, transformer models are the leading architectural basis for building context-sensitive language representations. Their use has radically changed approaches to analyzing emotions in texts, due to deep consideration of semantic, syntactic and pragmatic relationships between words. The most famous implementations of this architecture are the BERT (Bidirectional Encoder Representations from Transformers) [27] and GPT (Generative Pre-trained Transformer).

Transformer architecture is built on the mechanisms of multi-head self-attention which allow each word in a sentence to interact with all the others, assessing their relevance in context. This allows for the creation of vector representations that consider word order, grammatical dependencies, and latent semantics at all levels.

BERT is a bidirectional text encoding model that analyzes the context both to the left and to the right of the current word, which is critical for recognizing affectively

colored phrases. The main feature of such a model is bidirectionality, i.e. considering the full context of the sentence. First, the model is trained on large text arrays and then retrained for a specific task (emotion classification, stress recognition, etc.). Another feature of BERT is the use of a masking mechanism, which allows the model to "predict" missed words and learn dependencies between them.

When working with the Ukrainian language, it is advisable to use UkrBERT, developed by the UA NLP community, Multilingual BERT (mBERT) — a multilingual model that supports Ukrainian, and XLM- RoBERTa, a multilingual transformer model with extended support for Ukrainian vocabulary.

Due to transformer architecture, BERT has many advantages in emotional analysis in recognizing context-dependent polysemy (e.g., the word "strongly" in different emotional contexts). The features of self-learning and masking mechanisms ensure accurate detection of emotional markers even with a complex syntactic structure. It is also possible to visualize the attention matrix to identify key affective words.

Example. Fragment of a word representation vector using BERT (Figure 15).

```
BERT Embedding (UA - 'Я відчуваю тривогу завжди'):
[ 5.80473065e-01 -5.59855253e-02  1.03462553e+00  1.41433671e-01
 -4.54379052e-01  4.22969609e-01  8.42763707e-02 -4.60146228e-03
 -4.19105649e-01 -2.88306653e-01  1.14214495e-01 -8.01465437e-02
  1.74517810e-01 -4.19756323e-01  4.65718865e-01 -5.77060103e-01
  7.37607360e-01 -2.96049416e-01 -4.99203235e-01  3.00471038e-01
  6.91122338e-02  5.99023640e-01 -5.72020888e-01  1.45142615e-01
  2.81891435e-01 -5.90738691e-02  9.91712362e-02 -4.54924226e-01
 -2.20228538e-01 -6.26617014e-01  3.54803503e-01  3.00550014e-01
  1.03022940e-01  8.20589125e-01 -2.60571361e-01  2.37348944e-01
  3.65778148e-01  2.92586207e-01 -1.08028099e-01  1.93298563e-01
 -1.99231848e-01  4.87259850e-02  5.87955751e-02 -3.86342019e-01
  3.52562070e-01 -3.82899970e-01  1.22537637e+00 -1.50593193e-02
 -5.00051558e-01 -6.63393319e-01 -1.08061127e-01  4.53112036e-01
 -2.33623847e-01 -5.04290015e-02 -2.57973641e-01  1.20913245e-01
 -6.33411825e-01  5.51278107e-02  2.66463608e-01 -1.51190102e-01
  5.07499397e-01  5.01369298e-01  8.54027048e-02 -4.62225139e-01
 -4.99950320e-01 -3.81601363e-01  6.83535561e-02  6.25719428e-02
  3.69886488e-01  7.17610836e-01 -6.20892085e-02  1.06136091e-01
 -1.77784219e-01  5.06498180e-02  5.09840362e-02  8.26441944e-01
  6.78672194e-01 -2.32867628e-01 -1.63071856e-01 -4.04182315e-01
  8.75850469e-02  1.25886619e+00 -2.32907310e-01  5.35420626e-02
  6.53811544e-02 -1.12095229e-01  3.33286315e-01  2.05705732e-01
```

Figure 15 – Fragment of a word representation vector using BERT

GPT is an autoaggregative a transformative architecture that generates text sequentially, predicting each subsequent word. The model has a unidirectional context,

focused only on the previous words. GPT works as a language generator, for example, with the aim of generating emotionally relevant responses, it is used for dialog systems, text interpreters, paraphrasing. This model demonstrates significant efficiency in generating emotionally relevant descriptions or modeling reactions to stressful events.

In emotional analysis tasks, GPT performs classification using prompt-based instructions. learning), performs emotional summarization and generates texts for training emotional classification models.

3.2. Topic modeling of the texts

Topic modeling is a way of building a model of a collection of text documents that determines which topics each document belongs to.

The topic model of a collection of text documents determines which topics each document belongs to and which words (terms) form each topic.

The transition from the space of terms to the space of found topics helps to resolve synonymy and polysemy of terms, as well as more effectively solve such tasks as topic search, classification, summarization, and annotation of document collections and news streams.

Topic modeling as a type of statistical models for finding hidden topics found in a collection of documents has found its application in such areas as machine learning and natural language processing. Researchers use various topic models to analyze texts, text archives of documents, to analyze the change of topics in sets of documents. Intuitively understanding that a document refers to a certain topic, in documents devoted to one topic, you can find some words more often than others. For example: “dog” and “bone” are found more often in documents about dogs, “cats” and “milk” will be found in documents about kittens, the prepositions “and” and “in” will be found in both topics. Usually a document refers to several topics in different proportions, so a document in which 10% of the topic is cats, and 90% of the topic is dogs, we can assume that there are 9 times more words about dogs. Topic modeling reflects this intuition in a mathematical structure that allows us to conclude, based on studying a

collection of documents and studying the frequency characteristics of words in each document, that each document is a certain balance of topics.

The most widely used approaches in modern applications are those based on Bayesian networks − probabilistic models on directed graphs. Probabilistic topic models are a relatively young field of research in the theory of self-learning. One of the first was the probabilistic latent semantic analysis (PLSA), based on the principle of maximum likelihood, as an alternative to classical clustering methods based on the calculation of distance functions. Following PLSA, the latent Dirichlet placement method and its numerous generalizations were proposed.

Probabilistic topic models implement "soft" clustering, allowing a document or term to belong to several topics at once with different possibilities. Probabilistic topic models describe each topic as a discrete distribution over a set of terms, each document as a discrete distribution over a set of topics. It is assumed that a collection of documents is a sequence of terms chosen randomly and independently from a mixture of such distributions, and the task is to recover the components of the mixture from the sample.

Although topic modeling has traditionally been described and applied in natural language processing, it has also found applications in other fields, such as bioinformatics.

A topic model is a type of algorithm that scans a set of documents (a corpus), examines how words and phrases occur together in them, and automatically "learns" groups or clusters [86] of words that best characterize those documents. These sets of words often represent a consistent topic or theme.

There are many common topic modeling algorithms, including non-negative matrix factorization, latent Dirichlet distribution (LDA), and structural topic models. Ongoing research has also yielded other options. At the Pew Research Center Research Center, we tested some of these algorithms and explored how topic models can help analyze large collections of text, such as open-ended survey responses and social media posts.

Difference between topic modeling and clustering

Topic modeling is a statistical method for discovering hidden themes in a collection of documents. Clustering is a machine learning method for grouping similar data points. Both methods group documents, but they differ in how they do it.

Clustering algorithms group similar elements, while topic modeling algorithms identify relationships between elements. Topic modeling uses a statistical approach to find hidden themes in a collection of documents.

Clustering is commonly used to group elements together so that they can be analyzed.

Topic modeling finds relationships between elements and understands the hidden structure of a dataset. In addition, topic modeling does not require manual labeling of data points, while clustering requires manual labeling of data points. The figure 16 illustrates how topic modeling and clustering work.

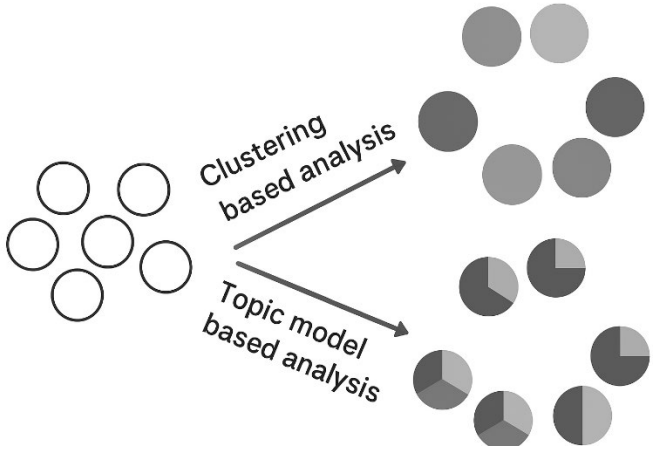


Figure 16 – Difference between topic modeling and clustering work

Topic modeling and topic classification

Both topic modeling and topic classification are unsupervised learning methods. This means that they do not require labeled data for training. Both can be used to discover hidden patterns in data, but they do so in different ways. Below is a comparison table 14 of topic modeling and classification.

Table 14 – The comparison of topic modeling and classification

Characteristic	Modeling topics	Topic classification
Type of approach	A more general unsupervised approach that is used for any type of data	A more specific, controlled approach that classifies documents according to predefined topics
Appointment	Reveals hidden topics in a collection of documents	Assigns a label to a document according to its topic

Topic modeling and text classification

While topic modeling involves searching for topics in a collection of documents, text classification uses text classifiers to assign a label to a document based on its content. Text classification is more specific and classifies documents into predefined categories.

Table 15 lists some key differences between topic modeling and text classification.

Table 15 – The key differences between topic modeling and text classification

Characteristic	Modeling topics	Text classification
Type of training	Uses unsupervised learning, meaning it does not require labeled training data	Uses supervised learning with machine learning techniques and labeled training data to train classification
Appointment	Can find hidden structures in text data	Focused on assigning labels to texts
Flexibility	More flexible as it does not require a predefined set of categories, which can be time-consuming and expensive without the right tools	Requires pre-labeling of data

Although topic modeling and text classification work differently, they can work together to produce the best AI predictions.

Basic models for topic modeling

Latent Dirichlet Allocation is one of the most widely used and conceptually validated methods of thematic modeling of texts, based on a probabilistic (Bayesian)

approach to detecting latent topics. The main idea of the algorithm is that each document is a mixture of several topics, and each topic is a distribution of terms with a certain probability. Thus, LDA allows you to model the incoming structure of a text corpus, detecting topics based on the co-distribution of words in documents.

LDA assumes that each document has a hidden topic distribution specific to it. Each topic is a word distribution that is common to all documents, and each word in a document is generated by a hidden topic that is selected from the document based on its topic distribution, and then a term is selected according to the word distribution of this topic.

Mathematically, building an LDA model involves several steps:

1. Initialization: each word is randomly assigned to one of the topics from a set of *k*, where *k* is the given number of topics.
2. For each word *w* in document *d*, the following is evaluated:
    - *P(t|d)* is the probability that a word in a document belongs to topic t, which is calculated as the fraction of words in the document that are assigned to that topic;
    - *P(w|t)* is the probability that word *w* comes from topic *t*, determined based on the frequency distribution of words in the topic.

These probabilities are combined to estimate the probability that a word belongs to a particular topic:

$$P(w\ with\ t) = P(t|d) \cdot P(w|t) \quad (8)$$

After several iterations, the model stabilizes, identifying topics and their representation in each document.

The advantages of LDA are interpretability, since topics are represented as lists of terms with corresponding weights, which provides transparency for humans. Each word and document can belong to several topics, which correspond to the realistic multi-topic nature of texts. The model also allows the simultaneous representation of a document through several thematic components. Another positive aspect is the wide software support, since implementations are available in Gensim, scikit-learn, MALLET, Spark, Stata, R. LDA supports the processing of large corpora, in

distributed environments.

Regarding the disadvantages of LDA, we can highlight its low efficiency on short texts, since short fragments do not contain enough information for a reliable assessment of topics. The model does not consider polysemy or homonymy (for example, "apple" as a fruit or a company). The use of the "bag of words" approach contributes to ignoring the order of words in a sentence. Determining the optimal value of k can be difficult and requires additional metrics (coherence, perplexity). LDA has low performance compared to methods based on matrix decomposition.

In the context of emotional or stress analysis, LDA can be used to identify thematic clusters in user texts, groups of words that indicate emotional states (anxiety, anger, loneliness, etc.) without prior labeling. Thus, the algorithm allows structuring corpora of appeals, comments, or diary entries with a high level of generalization.

Non-Negative Matrix Factorization (NMF). One of the leading thematic modeling methods in text analysis tasks is Non-Negative Matrix Factorization (NMF) is an algorithm that allows for the detection of latent themes in a corpus of documents through matrix decomposition with the constraint that all values are non-negativity. This provides high interpretability of themes, which is especially relevant for the analysis of emotionally colored or stressful texts.

The NMF algorithm involves decomposing the document-term matrix $X \in R^{m \times n}$, where m is the number of documents and n is the number of terms, into two lower dimensions of the matrix. $W \in R^{m \times k}$, and $H \in R^{k \times n}$, where k is the number of topics:

$$X \approx W \cdot H, \quad W, H \geq 0, \tag{9}$$

where W is the matrix of document affiliation to topics, H — matrix of word weights in topics.

All W and H elements are non-negative, which avoids negative weights and allows for direct interpretation of themes through the positive weights of key terms.

Stages of implementing NMF for thematic analysis are given below.

1. Building a TF-IDF matrix. An input corpus is used where each document is represented as a vector of weighted terms (TF-IDF). This reduces the weight of frequent terms and increases the significance of rare, potentially emotional words.

2. Decomposition into W and H The NMF algorithm is applied to the resulting matrix, forming topics as linear combinations of terms and documents as combinations of topics.

3. Topic interpretation. For each topic, top terms are determined by the largest values in the matrix H. This allows us to consider the topic as a semantic core of the most representative words.

Table 16 – Example of input matrix structure

Document	$word_1$	$word_2$	…	$word_n$
D_1	0.15	0.00	…	0.21
D_2	0.00	0.13	…	0.09
…	…	…	…	…

After the decomposition, *W* reflects which topics are dominant in each document, and *H* indicates which words are most characteristic of each topic.

The advantages of using NMF are the ability to represent a topic as a set of words with positive weights. The ability to use TF-IDF increases accuracy on short or informative texts. This method is easy to implement, as the scikit-learn library has a ready-made NMF implementation with a user-friendly interface. NMF is also suitable for small and medium-sized corpora.

However, despite its advantages, NMF has a number of limitations. For example, the results can significantly depend on the choice of parameter *k.* (number of topics). Another disadvantage is the lack of deep context accounting, as NMF does not consider word order or syntactic relations, unlike transform models (e.g., BERT).

The NMF method can be effectively used to identify emotionally charged topics in texts, especially when analyzing open-ended responses from questionnaires, anonymous complaints or appeals, posts from support forums, etc. For example, NMF can identify topics related to anxiety, fatigue, conflicts, loneliness, even without explicit text labeling.

BERTopic is an innovative topic modeling technique that combines transform models, such as BERT or XLM-R, with clustering algorithms, such as HDBSCAN,

and a class-based TF-IDF (c-TF-IDF) approach. The model ensures interpretability of topics by preserving the weight characteristics of words within each cluster.

BERTopic's workflow is presented as follows:

1. Formation of vector representations of texts using multilingual BERT (mBERT, XLM-R), which allows considering contextual connections between words.

2. Space dimensionality reduction using UMAP, a nonlinear reduction method that provides a compact representation for clustering.

3. Vector clustering using HDBSCAN, dense clusters are formed automatically, without the need to determine their number.

4. Interpretation of topics through c-TF-IDF, which allows identifying the most characteristic terms for each cluster of documents.

The advantages of BERTopic are support for the Ukrainian language thanks to multilingual transformers, the ability to detect semantic relationships, synonymy, and contextual meanings, automatic determination of the number of topics, and convenient integration into machine and hybrid text processing pipelines. BERTopic flow is shown on Figure 17.

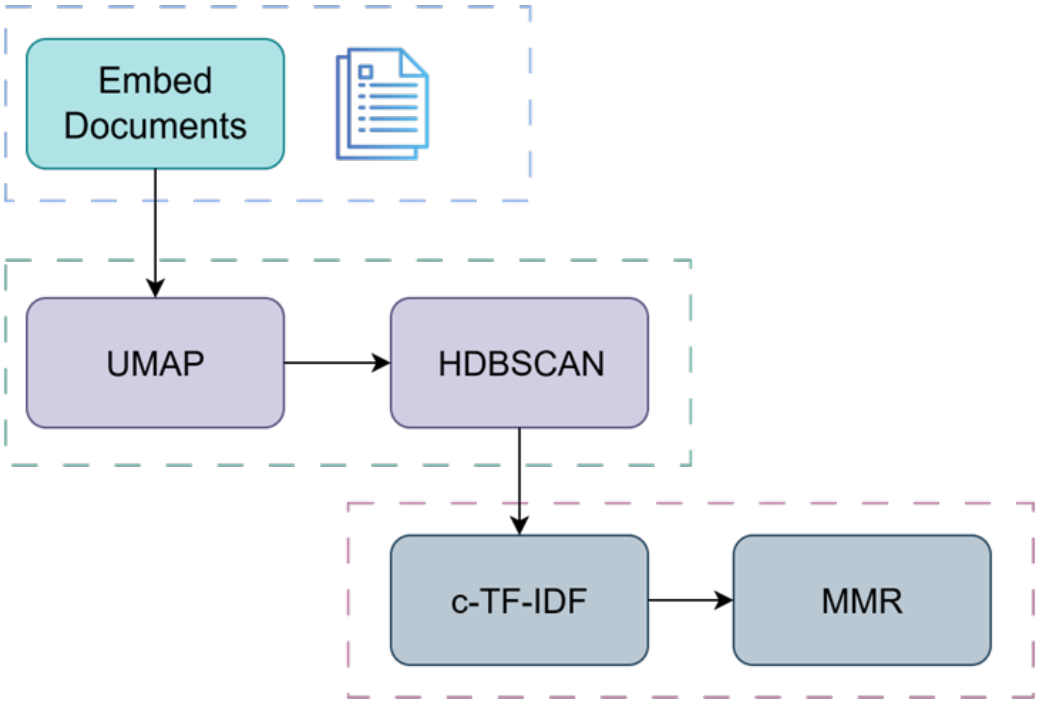


Figure 17 – BERTopic flow

Top2Vec is another modern approach to topic modeling that automatically detects topics by combining semantic vectors of texts and words with clustering methods. The method is based on the philosophy of "deep" topic modeling, where vector representations of documents and terms have the same space.

Top2Vec works based on forming vectors of documents and terms using Universal -type models. Sentence Encoder, Word2Vec or transformers. Next, the dimensionality of the vector space is reduced via UMAP. The next stage is clustering based on HDBSCAN, which allows you to automatically detect the number of topics. After that, marker words are selected for each cluster as thematic descriptors.

The features of Top2Vec are that this model does not require prior determination of the number of topics, demonstrates high processing speed without significant need for hyperparametric optimization. Top2Vec provides visualization of the topic space in 2D or 3D. This model is also suitable for large text corpora, both structured and informal.

Top2Vec is ideal for analyzing user reviews, requests, and forum posts, where the topic is dynamic and the structure is informal.

Latent Semantic Analysis (LSA), also known as Latent Semantic Indexing (LSI), is an early statistical method of topic modeling used to uncover hidden semantic structures in large text corpora. This approach is not probabilistic, like some modern methods, but is based on algebraic matrix analysis, in particular singular value decomposition (SVD). The basic idea is to reduce a large and sparse matrix of term frequency representations in documents to a lower semantic space with fewer dimensions.

To reduce dimensionality and identify latent semantic themes, SVD decomposition is used (10):

$$X \approx U_k \cdot S_k \cdot V_k^{\top}, \tag{10}$$

where $X \in R^{m \times n}$ initial matrix "document × term", $U_k \in R^{m \times k}$ matrix of documents in the topic space, $S_k \in R^{k \times k}$ diagonal matrix of stored singular values, $V_k^{\top} \in R^{k \times n}$ matrix of terms in the topic space, $k \ll \min(m,n)$ is the selected number of topics (the size of the latent space).

To implement this approach, a "document × term" matrix is first formed, in which each value is determined using TF-IDF, which allows us to consider both the frequency of words and their informativeness in the context of the entire corpus. After that, a truncated SVD decomposition is applied, resulting in three matrices: the first describes the projection of documents in a new thematic space, the second contains the scale coefficients of singular values, and the third reflects the terms in the same latent space. This allows us to represent each document or word as a vector of a certain dimension, reflecting its semantic proximity to other documents or terms.

Despite the effectiveness of this approach, especially for clustering, retrieval, or semantic aggregation of documents, LSA has several limitations. One of the main drawbacks is the difficulty of interpreting the resulting latent topics, since the vectors do not have a direct linguistic binding, like, for example, in probabilistic models. In addition, LSA is sensitive to the choice of the number of topics, does not consider the context of word use, and does not distinguish between word order in sentences. Therefore, it is less suitable for tasks where syntactic or contextual structure is important, for example, for detecting emotional connotation or polysemy.

Despite these limitations, LSA remains a powerful tool for preliminary thematic analysis, especially when it is necessary to quickly extract semantic groups of documents or terms in a large corpus. Its application can also be useful in the field of emotional analysis, for aggregating texts that share emotionally colored markers in latent space, where semantic clusters related to stress, anxiety, or other affective states are detected.

Probabilistic Latent Semantic Analysis (PLSA) is a method based on a Bayesian model that partitions terms in documents into topics. pLSA is an improvement on LSA, and it is a generative model that aims to find hidden topics in documents by replacing SVD in LSA with a probabilistic model.

PLSA uses the em -algorithm to find the distribution of terms in each topic. PLSA or probabilistic latent semantic analysis is a technique used to model information in a probabilistic structure. Latent because the topics are treated as hidden or latent variables.

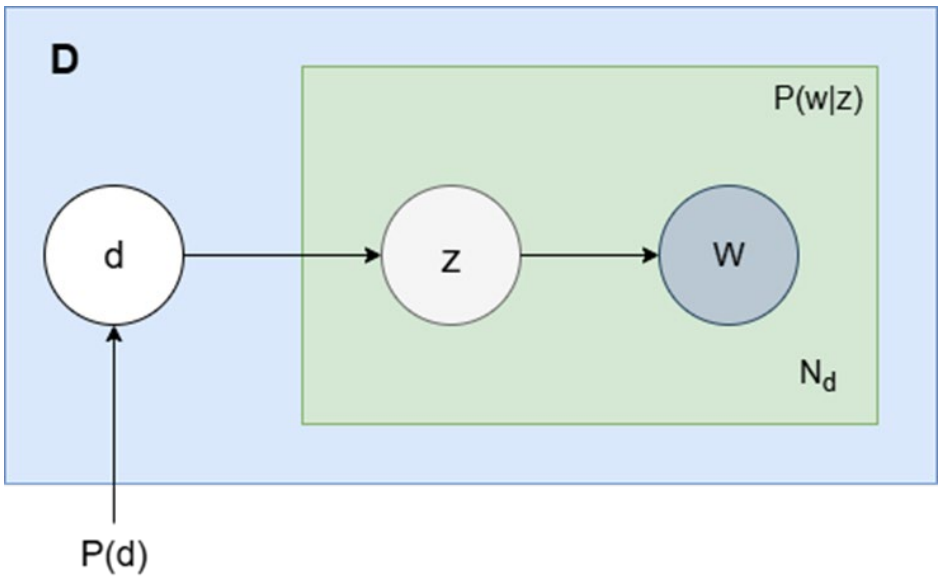


Figure 18 – Probabilistic Latent Semantic Analysis flow

PLSA can be understood in two different ways:

1) latent variable model;
2) matrix factorization.

The first method will help you understand the mathematics of PLSA very well. While the second method is easy to implement in Python.

Correlated Topic Models is an extension of the classical model that removes one of the main limitations of LDA, namely the assumption of topic independence. Unlike LDA, where topics are modeled as independent components using the Dirichlet distribution, CTM allows us to consider dependencies between topics by modeling their covariance structures. This approach better reflects real-world text data, where topics are often interrelated, for example, the topics "health" and "stress", "work" and "burnout", "war" and "fear" can appear simultaneously in the same document and have a positive correlation [62].

The basic idea of CTM is to use a logistic normal distribution instead of a Dirichlet distribution to model thematic proportions of a document. This allows topics to have an arbitrary covariance matrix, i.e. to model both positive and negative relationships between topics. Each document is represented as a point in the topic space, the coordinates of which are correlated with each other according to empirically identified relationships between topics. The figure below shows a schematic of how CTM works.

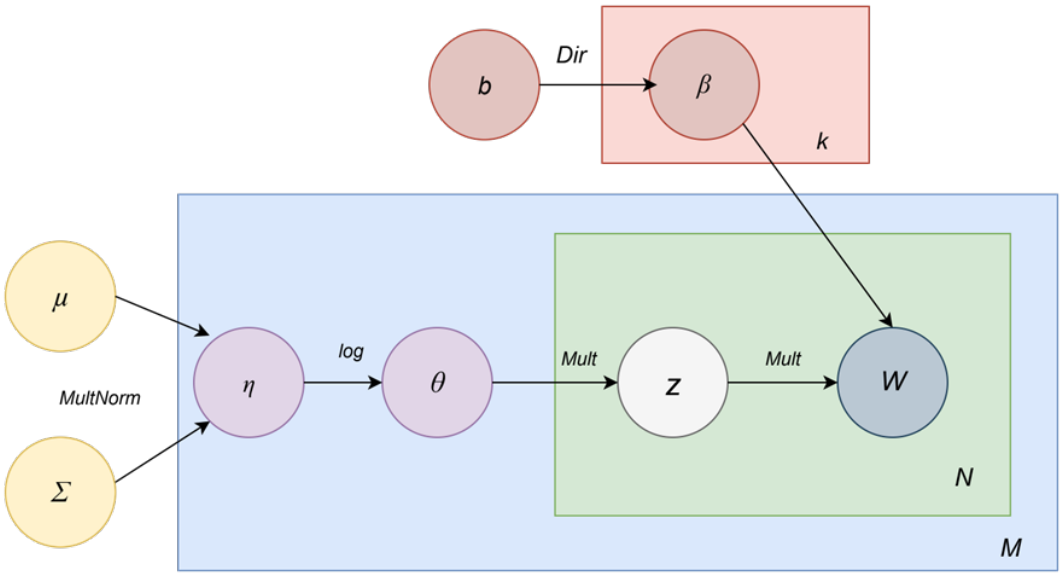


Figure 19 – CTM diagram

From a formal point of view, in CTM thematic proportions are generated from a multivariate normal distribution in log -space, which after normalization gives a vector of thematic weights. Then, as in LDA, a topic is chosen for each word according to this distribution, and the word is generated from the corresponding thematic distribution of terms.

Correlated Benefits Topic Models is the ability to detect co-occurring topics that appear simultaneously in many documents, a more flexible thematic representation of documents, which is useful for interpretation, classification, and visualization. Another significant advantage is the improvement of the quality of thematic modeling in corpora with thematically similar but contextually different messages.

CTM is particularly effective for studying psychologically loaded texts, where emotional themes, such as “anxiety,” “loneliness,” “help,” may coexist with social or medical themes [6]. In such cases, identifying correlated themes helps to understand the complex structure of the author’s affective state and allows for more accurate models for predicting stress or emotional risk.

The CTM model is supported in the gensim, topicmodels (in R), and scikit-learn libraries by combining lognormal distributions with topic analysis. There are also implementations in PyMC and Stan for building Bayesian models with arbitrary

dependency structures.

Hierarchical Dirichlet Process is a Bayesian thematic modeling method that allows you to model an unlimited number of topics. HDP uses a Bayesian nonparametric modeling approach that allows you to automatically determine the number of topics based on the data.

HDP is an extension of LDA designed for the case where the number of components in the mixture (the number of "topics" in document modeling terms) is unknown a priori. When using LDA to model documents, one treats each "topic" as a distribution of words in some known dictionary. For each document, a mixture of topics is drawn from a Dirichlet distribution, and then each word in the document is an independent pattern from this mixture (i.e., a topic is selected and then used to generate a word).

HDP also uses the Dirichlet distribution to capture the uncertainty in the number of topics. So, a common base distribution is chosen that represents the infinite set of possible topics for the corpus, and then the final topic distribution for each document is chosen from this base distribution.

In terms of pros and cons, HDP has the advantage that the maximum number of topics can be unlimited and studied based on the data rather than specified in advance. This process is more difficult to implement and unnecessary in the case where a limited number of topics is acceptable.

The generating model for the hierarchical Dirichlet process of the latent Dirichlet distribution is as follows in (11):

$$\begin{gathered} H \sim Dirichlet(\beta) \\ G_0|\gamma, H \sim DP(\gamma, H) \\ G_j|\alpha_0, G_0, \sim DP(\alpha_0, G_0) \\ \theta_{ji}|G_j \sim G_j \\ x_{ij}|\theta_{ji} \sim Categorical(\theta_{ji}) \end{gathered} \quad (11)$$

where H is a Dirichlet distribution whose size is the size of the dictionary, i.e. it is a distribution over an uncountable number of term (topic) distributions, G_0 is a

distribution over a countably infinite number of distributions of categorical terms, i.e., topics. For each document j, G_j is a distribution over an infinite number of distributions of categorical terms, i.e., topics. θ_{ji} is a categorical division by terms, i.e. topic; x_{ij} is a term.

Structural Topic Model is a thematic modeling method that allows us to consider the structure of documents and the interactions between topics. STM uses multivariate linear regression to determine the influence of variables on topics and the relationships between topics.

One of the coolest things about topic modeling is that it has applications in a variety of fields. It can help motivate queries, provide unique insights into texts, and give you new ways to organize documents.

A Structural Topic Model (STM) is a form of topic modeling specifically designed with social science research in mind. STM allows us to incorporate metadata into our model and discover how different documents might talk about the same underlying topic using different wording variations.

Two documents may deal with the same topic, say, a protest, but approach the topic from different perspectives. Perhaps one tends to focus on "police brutality" or "peaceful protesters," while the other uses terms like "law and order" and "radical rioters." You could say that both documents deal with the same topic, but the content of the topic (i.e., the words that make up the topic) differs from document to document.

Both the prevalence of a topic and the content of a topic for a particular document correlate with "metadata" about the document. For example, certain sources may write about politics more often or write about politics in a certain way. Metadata may include publication date, author, publication, social media likes, or any number of categorical or numeric variables about the document.

STM not only allows you to create higher quality models, but also provides insights into the corpus, such as how metadata affects the words a document uses in a topic.

Dynamic Topic Models is a thematic modeling method that allows us to account for changes in topics over time. DTM uses a multidimensional version of LDA to detect

topics at each point in time and find relationships between topics at different times.

DTM is a set of techniques that aim to analyze the evolution of topics over time. These techniques allow us to understand how a topic is represented at different times. For example, in 1995, people might talk about environmental awareness differently than they do in 2015. Although the topic itself remains the same, environmental awareness, the exact representation of that topic may differ.

BERTopic allows us to use DTMs, computing a representation of a topic at each time step without having to run the entire model multiple times. To do this, we first need to fit BERTopic as if there was no temporal aspect to the data. This will create a general topic model. We use a global representation of the main topics that are likely to be found at different time steps. For each topic and time step, we compute a c-TF-IDF representation. This will result in a representation of a specific topic at each time step without having to create clusters from already created built-in components (Figure 20).

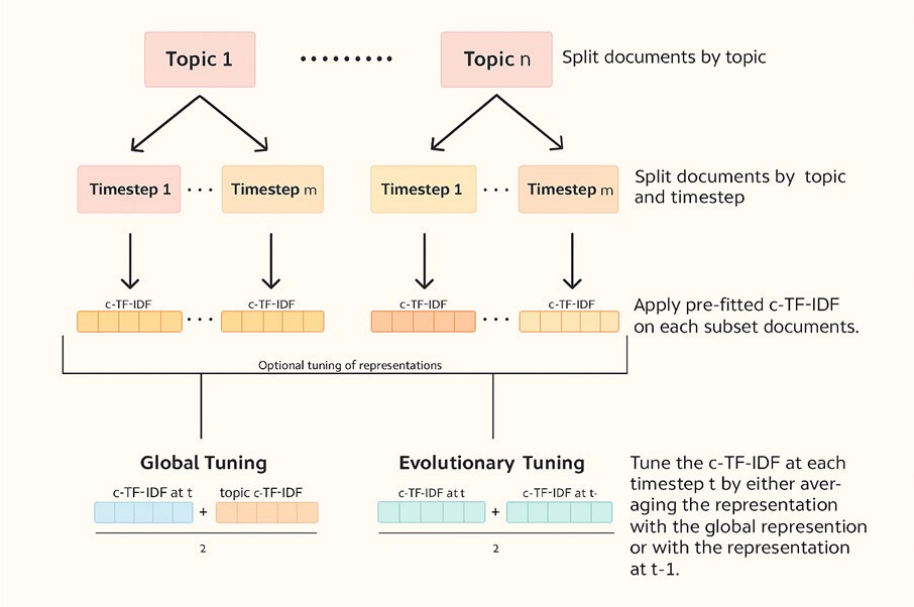


Figure 20 - Dynamic Topic Models flow

Next, there are two main ways to further fine-tune these specific topic representations, namely global and evolutionary.

The representation of a topic at time step *t* can be adjusted globally by averaging its c-TF-IDF representation with that of the global representation. This allows each

topic to move slightly towards the global representation while retaining some of its specific words.

The topic representation at time step t can be evolutionarily tuned by averaging its c-TF-IDF representation with the c-TF-IDF representation at time step *t-1*. This is done for each topic representation, allowing the representations to evolve over time.

Both fine-tuning methods are set to True by default and allow you to create interesting views.

Within the emotional analysis of texts, the study of the correlation between the thematic content of messages and the presence of stress markers deserves special attention. The idea is that certain topics raised in texts can be statistically associated with an increased level of psychological tension, anxiety or frustration of the author. Thematic modeling, based on the LDA, BERTopic or Top2Vec methods, allows you to automatically identify groups of semantically similar texts, which can then be analyzed in terms of the presence of stressful emotional patterns.

In this type of study, each topic obtained as a result of modeling is described by a list of the most relevant terms or top words, which allows it to be interpreted. For example, topics associated with words such as “anxiety”, “war”, “loss”, “separation”, “fear” can with a high probability signal the presence of a stressful state in the author. Further comparison of the thematic affiliation of the document with its emotional markup (stress/ neutral/undefined) allows us to statistically assess the strength of the connection between certain topics and the level of stress.

Correlation analysis can be performed using metrics such as Pearson's or Spearman's correlation coefficients, which allow quantitative measurement of the relationship between the intensity of a document's belonging to a certain topic (the vector weights of topics in the model) and the probability of detecting stress markers in the text. It is also possible to use cluster analysis to identify groups of topics associated with increased emotional stress, as well as classification methods (for example, SVM or Random Forest), which use thematic features as predictors.

Thus, correlation analysis between topics and stressful emotions serves as an important tool in building psycholinguistic profiles of users, early detection of crisis

messages in social networks, as well as in modeling a person's emotional state based on their written discourse.

3.3. Sentiment analysis

Sentiment analysis [90], tonality analysis, also known as opinion analysis, sentiment analysis, is a natural language processing approach that determines the emotional tone [92] behind text. It is a popular way for organizations to identify and classify opinions about a product, service, or idea. Sentiment analysis involves the use of data mining, machine learning, and artificial intelligence [63] to find sentiment and subjective information in text.

Sentiment analysis systems help organizations gather information from unstructured and unstructured text that comes from online sources such as emails, blog posts, support tickets, web chats, social media channels, forums, and comments. Algorithms replace manual data processing by implementing automatic or hybrid rule-based methods. Rule-based systems perform sentiment analysis based on predefined lexicon-based rules, while automatic systems learn from data using machine learning techniques. Hybrid sentiment analysis combines both approaches.

In addition to detecting sentiment, sentiment analysis can highlight the polarity (or amount of positivity and negativity), topic, and supporter of an opinion [68] in a text. In addition, sentiment analysis can be applied to different domains, such as the document, paragraph, sentence, and sub-topic levels.

Sentiment analysis platforms or SaaS products include Brandwatch, Hootsuite, Lexalytics, NetBase, Sprout Social, Sysomos, and Zoho. Businesses that use these tools can review customer feedback more frequently and respond early to changing market sentiment.

Types of sentiment analysis are described below.

1. *Fine-grained* sentiment analysis provides a more precise level of polarity, breaking it down into additional categories, typically from very positive to very negative. This can be thought of as the equivalent of a 5-star rating scale.

2. *Emotion recognition* identifies specific emotions, not positive or negative. Examples might include happiness, disappointment, shock, anger, and sadness.

3. *Intent-based analysis* recognizes the actions behind text, in addition to thoughts. For example, an online comment expressing frustration about a battery replacement may prompt customer service to reach out to resolve that specific issue.

4. *Aspect analysis* captures a specific component that is mentioned positively or negatively. For example, a customer may leave a review about a product, noting that the battery life is too short. The system will then report that the negative sentiment is not about the product as a unit, but about the battery life.

Let's consider in more detail the features of opinion analysis.

In the basic case, machine learning models classify text into one of three categories – positive, negative, neutral tone.

Example. The sentences below show three categories of tone [93].

- Positive: “The service was excellent, everything was fast and convenient”;
- Negative: “It was the worst trip of my life”;
- Neutral: “The train arrived at 3:30 p.m. according to schedule”.

Additional categories when determining tone can be sarcasm and irony, when the meaning of an expression contradicts its literal sound.

Example. In the sentence “Great! Another road accident, just what I was waiting for today.” there is positive vocabulary here like word “great”, but negative emotion → a special model is needed to detect sarcasm.

For further expansion, emotional coloring categories can be added, meaning the tone can be detailed by specific emotions:

- joy
- anger
- fear
- sadness
- surprise
- disgust

This allows us not only to say whether the text is “positive” or “negative”, but

also to understand what feeling the author is expressing.

Example. The sentences below show different emotions in texts.

- Joy: "Finally, I'm on vacation! I'm thrilled!";
- Anger: "I hate it when someone is an hour late.";
- Fear: "I don't know what to do, I'm scared for tomorrow.".

Another way to classify emotions is to look for polarity and intensity.

Polarity describes the direction of an emotion:

- Positive polarity;
- Negative polarity;
- Neutral/

Polarity = sign (e.g., +1, -1, 0)

Intensity or strength of an emotion shows how strongly the emotion is expressed. It is measured, for example, from 0 to 1 (or from -1 to +1). It allows you to distinguish between weak and strong positive: "good" (weak positive), "great", "incredible" (strong positive).

Example. Different examples of polarity and intensity of emotions in Ukrainian texts are shownt in table 17.

Table 17 – Different examples of emotions polarity and intensity in Ukrainian texts

Text	Polarity	Intensity
"Все нормально."	neutral	low
"Я щасливий!"	positive	high
"Мені огидно навіть думати про це."	negative	very high

Tone analysis is not limited to just "+" or "–". Modern models consider context, intensity, sarcasm, specific emotions. This makes the task more interesting, but also more difficult for automatic analysis.

Stress emotions are a subgroup of emotions with a predominantly negative tone, which are accompanied by psycho-emotional tension. These include anxiety, fear, helplessness, overload, emotional exhaustion. Although they have a negative tone, not

all negative emotions are stressful. For example, anger can be an expression of a reaction to stress but is not always an indicator of a destructive state. That is why detecting stress requires a combined approach, i.e. not only classification by polarity, but also recognition of the emotional category and contextual modeling.

Among the methods of tonal analysis are:

1. Dictionary approaches, where dictionaries of emotional terms are used (e.g. SentiWordNet or adapted Ukrainian-language lexicons), and each word has a polarity score. In this approach, the total tone of the text is calculated through the aggregation of values [97].

2. Statistical methods based on the use of the TF-IDF measure with weighting coefficients for emotional words. Linear regression and logistic regression are often used to predict the tone class [96].

3. Contextual models based on transformers [95]. The use of BERT, XLM-R, UkrBERT models is suitable for classifying tone considering deep context. Such models can identify complex structures, irony, negation, and emotionally loaded syntactic constructions.

To detect stress, tone analysis acts as a first-level filter. In general, the algorithm works as follows:

1. The polarity of the text (positive/negative/neutral) is determined.
2. If the tone is negative, then:
    - An additional analysis of the type of emotion is performed, for example, “anxiety”, “exhaustion”, “powerlessness”.
    - An assessment of lexical and grammatical patterns typical of stressful situations is performed.
    - Intensity is identified using amplifiers, for example, “I’m very scared,” “I’m terribly tired.”

Thus, emotionally negative statements [98] can be further classified as stressful or non-stressful.

The analysis of the tone of texts is an effective tool for the basic determination of the emotional background of the text. In the tasks of stress detection, it is used as a

preliminary filter that allows narrowing the range of potentially emotionally saturated statements [100]. In combination with linguistic and semantic characteristics, the tone analysis becomes a key component of a comprehensive system of emotional analysis of Ukrainian-language texts.

3.4. Identifying signs of stress

Stress identification in Ukrainian texts involves the identification of linguistic markers that characterize cognitive-emotional tension, psycholinguistic instability, and affective shifts. Unlike general tonality analysis, stress pattern [77] identification requires the integration of several linguistic levels, namely lexical, grammatical, and syntactic.

Stressful states are often expressed through stable phraseological or syntactic structures that contain:

- emotionally charged verbs, for example, *"боюсь", "не витримую", "дратує", "тривожуся"*;
- evaluative adjectives, for example, *"жахливий", "нестерпний", "поганий", "втомлений";*
- negation and uncertainty, for example, "*не знаю", "не впевнений", "ніяк не можу"*;
- exclamations and elliptical constructions, for example, *"Жах!", "Невже це кінець?", "Я більше не можу…".*

Such lexical patterns can be formalized as regular expressions or templates for automatic detection.

The frequency and repetition of lexical patterns can also indicate a stressful tone of the text. Excessive repetition of emotionally marked words, for example, *"боюся, боюся, боюся"* is a sign of affective fixation. Frequent use of first-person singular pronouns, for example, *"я", "мене"* demonstrates a focus on one's own experience. And a high frequency of modal verbs, for example, *"не можу", "мусив би"* may be evidence of psychological pressure or conflict.

Stressful utterances can be accompanied by both simplification of grammatical structure and its complication (excessive nesting of subordinate clauses). Short, fragmentary sentences, such as *"Не можу. Все погано. Жити важко."* are characteristic of low grammatical complexity. On the other hand, high grammatical complexity, i.e. the presence of long sentences with several subordinate clauses, indicates cognitive overload, for example, *"Я намагаюся зосередитися, але думки плутаються, і здається, що все це марно."*.

Syntactic shifts indicate an emotional deviation from the neutral structure:

- inversion: *"Не витримую я вже цього"* instead of *"Я вже не витримую цього"*;
- discontinuity of constructions: *"Я... не знаю, мабуть..."*;
- repeat with increasing intensity: *"Я боюсь. Дуже боюсь. Боюсь постійно."*;
- exclamations and pauses: *"Біль… Страх… Самотність…"*.

To detect stress features in a text, you can use an algorithm where the input data will be texts in Ukrainian, and the result of the algorithm will be the label stress or neutral pattern. Below is the sequence of steps of such an algorithm.

Step 1. Text preprocessing. At this stage, tokenization, lemmatization, stemming, part-of-speech extraction, etc. are performed.

Step 2. Identifying linguistic patterns. A search is performed for keywords from the emotion dictionary (fear, anxiety, panic, etc.). Pattern recognition is also performed, for example, "не можу + інфінітив", "я боюсь", "мені важко" etc.

Step 3. Frequency analysis. At this stage, the frequency of personal pronouns (1st person), modal verbs, and negative constructions is calculated. A threshold assessment based on empirical data is also performed.

Step 4. Grammatical and syntactic analysis. The dependency tree is built, which helps detect excessive nesting, exclamations and ellipsis, and violations of neutral word order.

Step 5. Comprehensive assessment. The feature vector is calculated, after which the data is passed to a classifier, the result of which will be the designation of features

as stressful or neutral.

The integration of linguistic features allows for accurate detection of stressed utterances. The proposed algorithm is based on a combination of rules, statistics, and syntactic analysis, which ensures interpretability and adaptability to Ukrainian language corpora.

Conclusions

Converting text documents into vector elements is a mandatory step for further analysis. Common classifiers and machine learning algorithms cannot directly process text in its original form. Documents must be converted into vector representations (features), where each feature is a dimension in feature space. The weight of these features can vary from a simple binary representation, i.e., the presence/absence of a word, to complex weighting schemes that consider the frequency of words in the document and the entire collection.

There are several basic approaches to obtaining features from text, each with different advantages and disadvantages, the application of which depends on the task at hand and the specifics of the subsequent steps of intelligent text analysis.

Topical modeling is a powerful statistical method for identifying hidden topics in document collections. It helps solve problems of synonymy and polysemy, as well as perform topic search, classification, and referencing tasks more efficiently. Correlation analysis between topic content and stress markers is a key tool for detecting psychological stress. Certain topics, such as “anxiety,” “war,” and “loss,” can statistically signal stress. This allows you to build psycholinguistic profiles of users and identify crisis messages.

Sentiment analysis is a fundamental approach to natural language processing that determines the emotional tone of a text. It uses methods [99] of intelligent data analysis, machine learning, and artificial intelligence to search for moods and subjective information. In stress detection tasks, sentiment analysis functions as a first-level filter, where polarity is first determined. In the case of negative sentiment, additional analysis

of emotion type, lexical-grammatical patterns, and intensity is performed.

Identifying stress in Ukrainian-language texts requires the identification of specific linguistic markers at the lexical, grammatical, and syntactic levels. Lexical markers include emotionally charged verbs, evaluative adjectives, negations, and interjections. The frequency and repetition of lexical patterns are also indicators of stress, such as excessive repetition of emotional words, frequent use of first-person singular pronouns, and high frequency of modal verbs indicate stress. The grammatical complexity of sentences can vary, from simplified, fragmented sentences to complex, overly nested constructions, indicating cognitive overload. Syntactic shifts include inversion, broken constructions, repetitions with increasing intensity, interjections, and pauses. The proposed algorithm for detecting stress comprises five steps: preprocessing of the text, detection of linguistic patterns, frequency analysis, grammatical and syntactic analysis, and comprehensive evaluation with transfer of features to the classifier. This integrated approach ensures the accuracy and interpretability of stress detection.

In general, effective analysis of emotional markers and stress detection in Ukrainian-language texts requires a comprehensive approach that combines various methods of text representation, topic modeling, tone analysis, and the identification of specific linguistic features. This allows not only to identify the presence of stress, but also to understand its causes and context, which is critical for psycholinguistic profiling and monitoring of emotional state.

KAPITEL 4 / *CHAPTER 4*
METHODS OF INTELLECTUAL ANALYSIS OF THE TEXTS

4.1. Overview of classification methods

Learning curves of texts is one of the tasks of intelligent text analysis, which consists of assigning a document to one of several categories based on its content. Classification can be done manually or automatically, using a set of rules or machine learning methods.

Text classification should be distinguished from clustering. In the latter case, texts are also grouped according to certain criteria, but there are no predefined categories.

In other words, text classification (i.e., text categorization or text labeling) is the task of assigning a set of predefined categories to open text. Text classifiers can be used to organize, structure, and classify virtually any type of text, from documents, medical research [7], and files to the entire Internet. For example, new articles can be organized by topic; support tickets can be organized by urgency; chat conversations can be organized by language; brand mentions can be organized by sentiment; and so on.

Below is an example of how classification works.

Consider the following text: "The user interface is quite intuitive and easy to use". A text classifier can take this phrase as input, analyze its content, and then automatically assign appropriate tags, such as Interface and Intuitive or Easy to Use.

The main task of text classification

The general task of text categorization can be formally defined as the task of approximating an unknown category assignment function $F: D \times C \rightarrow \{0, 1\}$, where D is the set of all possible documents and C is the set of predefined categories. The value of F (d, c) is 1 if document d belongs to category c, and 0 if it does not.

An approximation function $M: D \times C \rightarrow \{0, 1\}$ is called a classifier, and the task is to create a classifier that gives results that are as close as possible to the true category of the assignment function F.

Table 18 – Classification corpus representation

	d_1	…	d_j	…	d_n
c_1	*{0, 1}*	…	*{0, 1}*	…	*{0, 1}*
…	…	…	…	…	…
c_i	*{0, 1}*	…	*{0, 1}*	…	*{0, 1}*
…	…	…	…	…	…
c_m	*{0, 1}*	…	*{0, 1}*	…	*{0, 1}*

Single-Label versus Multilabel Categorization

Classic machine learning methods, such as Naive Bayes, logistic regression, and SVM, are often used for text classification, including psycho-emotional state analysis. They work well on small and medium-sized datasets. Logistic regression and SVM show high accuracy in tone analysis tasks (this can be seen in articles []).

However, classical approaches have limitations in understanding context, semantics, and complex emotional dependencies in text. They cannot effectively take into account the connections between words and subtle nuances of language, such as sarcasm or changes in emotional tone in long texts.

Logistic regression is one of the most interpreted and effective classical machine learning methods for binary and multi-class classification. In the context of psycho-emotional state analysis, it is used to predict the probability of a text belonging to a certain category, such as “anxiety,” “obsessive-compulsive disorder,” or “depression.” The input text is first converted into numerical vectors using various techniques, such as TF-IDF, Word2Vec, and others, after which logistic regression calculates the weights for each feature and finds the most likely class.

Unlike complex models, such as neural networks, logistic regression works quickly and does not require significant computing resources, making it ideal for experimentation and implementation. For tasks of sentiment analysis and basic analysis of psycho-emotional state, a logistic model is often sufficient, since there are no very

complex relationships between characteristics in such tasks.

In fact, the above-mentioned advantages and disadvantages of this method can be explored in practice. Now let's take a closer look at the algorithm of logistic regression.

1. Find linear combinations between feature vectors X and weight matrix W.

$$Z = w^T x + b \quad (12)$$

x — feature matrix (size m×n, where m is the number of samples and n is the number of features),

w^T — weight matrix (size n×k, where k is the number of classes),

b – bias,

Z — a matrix containing the results for each class for each sample.

For each sample i (row of matrix Z), we calculate the probabilities of belonging to each class using the softmax function [8] on Figure 21:

$$P_i = softmax(Z_i) = \frac{exp(Z_i)}{\sum_{k=1}^{K} exp(Z_{ik})} \quad (13)$$

P_i — probability vector for the i-th sample, where the sum of all elements is equal to 1,

Z_i — vector of linear combinations for the i-th sample,

k — class index.

As a result, we obtain a matrix P of size m×k, where each element P_{ik} is the probability that the i-th sample belongs to class k.

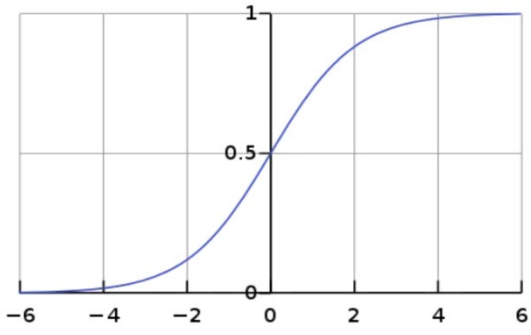


Figure 21 - Representation of the softmax function

1. For each sample i, we determine the class Y_i that has the highest probability P_{ik}.

2. Loss function (negative logarithmic likelihood)

To optimize, we choose the negative logarithm of likelihood, normalized by the number of samples:

$$J(W)=-\frac{1}{m}\sum_{i=1}^{m} log(P_{i,k=Y_i}) \quad (14)$$

J(W) — loss function (logarithmic loss, or Log Loss),

$P_{i,k=Y_i}$ — probability of correct class for the i-th sample.

Weight optimization (Gradient descent)

To minimize the loss function, we calculate the derivative with respect to weights W and update them:

$$W:=W-\alpha\frac{\partial J(W)}{\partial W}\partial J(W)\partial W \quad (15)$$

α — learning coefficient,

$\frac{\partial J(W)}{\partial W}$ — loss function gradient.

3. Training cycle

Initialize weights W with zeros or small random values.

Repeat:

- Calculate Z, P, loss function J(W),
- Update weights W using gradient descent.

Terminate iterations when the change in loss function J(W) becomes sufficiently small.

Comperison of Single-Label and Multilabel Categorization - Depending on the properties of the F function, we can distinguish between single-label and multilabel categorization. In multilabel categorization, categories overlap, and a document can belong to any number of categories. In single-label categories, each document belongs to exactly one category. Binary categorization is a special case of single-label categorization, for which there are two categories. Binary categorization is the most important because it is the simplest, most common, and most often used to demonstrate categorization methods. In addition, the general case with single-tag categorization is often a simple generalization of binary categorization. Multi-tag categorization can be solved using |C| binary classifiers (|C| is the number of categories), one for each

category, and each classifier decides whether to assign a document to different categories independently of each other.

Document-Pivoted versus Category-Pivoted Categorization - Classifiers are typically used as follows: upon receiving a document, the classifier identifies all categories to which the document belongs. This is referred to as document-oriented categorization. Alternatively, we may need to find all documents that should be submitted under a given category. This is referred to as category-oriented categorization. The difference is only significant when not all documents or categories are immediately available. For example, with online categorization, documents arrive one at a time, and therefore only document-based categorization is possible. On the other hand, if the set of categories is not fixed and if documents must be reclassified according to new categories, then we are dealing with category-oriented clustering. Most classification methods work with both approaches.

Hard and soft categorization - a fully automated categorization system makes a binary decision on each pair of document categories. Such a system is said to perform rigid categorization. However, the level of efficiency currently achieved by fully automated systems may be insufficient for some applications. In this case, a semi-automated approach is appropriate, in which the decision to assign a document to a category is made by a person, for whom the text classification system provides a list of categories ranked according to the system's assessment of the relevance of the category to the document. In this case, the system is considered to perform soft or rating categorization. Many classifiers actually have a whole segment [0, 1] in their range—that is, they generate a real value from zero to one for each category-document pair. This value is called the categorization status value (CSV). Such “continuous” classifiers naturally perform rating categorization, but if a binary decision is required, this can be done by checking the CSV against a certain threshold.

There are various possible methods for setting the threshold value. For some types of classifiers, threshold values can be calculated analytically using theoretical measures such as utility. There are also general methods that are independent of the classifier. The corrected threshold value assigns exactly k categories of the highest rating to each

document. A proportional threshold sets the threshold so that the same proportion of the test set belongs to the same category as the corresponding proportion of the training set. Finally, the most common method is to set the threshold value so as to maximize the performance of the classifier on the test set. A test set is a portion of the training set that is not used to create the model. The sole purpose of the test set is to optimize certain parameters of the classifier, such as the threshold.

Naive Bayes classifier. The Naive Bayes classifier is a machine learning algorithm for classifying text data based on Bayes' theorem and the assumption of independence of features (words) in documents (which is a "naive" assumption, hence the name).

In a naive Bayes classifier, each document (or text) is treated as a feature vector, where each feature represents the ratio of the number of occurrences of each word to the total number of words in the document. The algorithm stores pre-calculated probabilities of documents belonging to each class (category) and uses them to classify new documents.

The naive Bayes classifier assumes that all features (words) in a document are independent of each other, which is a simplification that does not always correspond to the real world, but allows the algorithm to be applied effectively to large amounts of data. Based on this assumption, the probability that a document belongs to a certain class is calculated by multiplying the probabilities of each word in the document belonging to that class.

The naive Bayes classifier is one of the simplest and most effective algorithms for text classification.

The general algorithm is shown below (Figure 22).

1. Data preparation. Select dataset + preprocessing + representation.
2. Model is trained, based on Bayes theorem

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}, \quad (16)$$

where P(C | X) is the probability of class C for a given text X, P(X|C) is the probability of words appearing in the text given class C, P(C) is the probability of each class in the sample, and P(X) is the total probability of words appearing.

3. Classification of new text. Breaking down new text into words + calculating the probability of belonging to each class + selecting the class with the highest probability.

The Naive Bayes algorithm learns quickly and makes predictions, making it extremely effective for large data sets and real-time applications. It is well suited for multi-class tasks, as it is very effective for classification tasks with multiple classes. It naturally handles multi-class problems and scales well when new classes are added. The algorithm works well even with limited training data, especially compared to other machine learning algorithms. It can achieve good results with a small number of training examples.

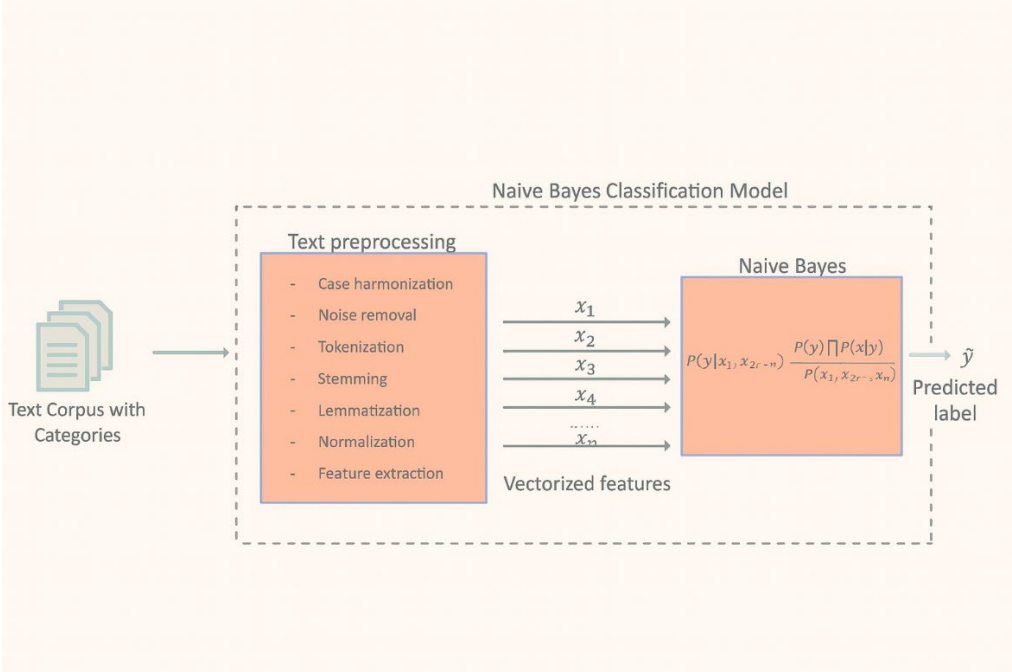


Figure 22 – Visualization of the classifier based on naive Bayes

The Naive Bayes algorithm learns quickly and makes predictions, making it extremely effective for large data sets and real-time applications. It is well suited for multi-class tasks, as it is very effective for classification tasks with multiple classes. It naturally handles multi-class problems and scales well when new classes are added. The algorithm works well even with limited training data, especially compared to other machine learning algorithms. It can achieve good results with a small number of training examples. Naive Bayes is particularly useful for categorical variables, such as

text classification using a bag-of-words model or spam filtering. The algorithm effectively predicts the class of test data, and you can quickly classify new data after training the model. If the assumption of feature independence is correct, Naive Bayes can be more accurate and faster than more complex algorithms such as logistic regression. If numerical data is used, the algorithm assumes that the features are distributed according to a Gaussian (normal) distribution, which is a reasonable assumption for many real-world datasets.

However, despite the numerous advantages of this classifier, Naive Bayes has a number of drawbacks. In particular, if a categorical feature in the test data was not observed during training, i.e., its value is missing in the training data, Naive Bayes assigns a zero probability to this class. This makes it difficult to predict the class correctly. This problem can be solved using Laplace smoothing or zero-frequency smoothing. Another major limitation is the assumption that all features are independent. In real-world datasets, this is rarely the case. Many features are interdependent, and this assumption can lead to suboptimal results when the features are highly correlated.

Since the algorithm assumes that all features have an equal and independent influence on the result, it may not work well in tasks where the interaction between features is important or where the relationships between features and classes are complex. Naive Bayes does not work very well with features that have complex relationships or require nonlinear models, such as images, speech, or deep learning tasks. It is more suitable for simpler tasks where the assumption of feature independence makes sense. Naive Bayes can be sensitive to irrelevant features in the dataset, which can reduce its effectiveness. If there are irrelevant features in the data, they can skew the predictions, making the model less reliable.

Expectation maximization (EM) algorithm − machine learning algorithm that is often used in text classification tasks. The EM algorithm is based on the idea that if we do not know the exact values of some model parameters, we can find them by performing successive iterations between two steps: the Expectation step and the Maximization step.

In text classification tasks, the EM algorithm can be used to estimate the parameters of the Bernoulli categorical model, which allows us to take into account the frequency of each word in a document relative to the total number of words in the document.

In the first step of the EM algorithm, the expectation step, we use the current values of the model parameters to determine the expected number of occurrences of each word in each of the categories (texts). Next, in the maximization step, we change the model parameters in such a way as to maximize the expected number of occurrences of each word in each of the categories.

The EM algorithm is used in text classification tasks because it can help in understanding which words or phrases best correspond to each category of texts and can help in creating a more accurate text classification model.

Example. Texts need to be classified into "Stressful" or "Non-stressful" classes in the absence of class labels for all texts, using the Expectation-Maximization method to find hidden classes. The input data is unlabeled texts, and it is not known which ones contain stressful context and which ones do not.

"Я не можу заснути вже третю ніч поспіль."

"Рівень води в Дніпрі цього року стабільний."

"Серце б'ється шалено, паніка не відпускає."

"Сьогодні науковці опублікували нове дослідження."

"Мене лякає кожен телефонний дзвінок."

"У Києві заплановано відкриття нової бібліотеки."

The first step is to randomly divide the texts into two classes. Class A: Texts 1, 3, 5. Class B: Texts 2, 4, 6. It is assumed that the words in each class have certain probabilities of occurring there.

Expectation (E-step) – Probability assessment. An analysis of the frequency of words in each group is performed. For example, in group A, the words "can't," "panic," "won't let go," "scares," and "call" predominate, indicating psychological tension or anxiety. In class B, the frequent words are "scientists," "stable," "research," "discovery," and "library" — informative and unemotional.

Then, for each text, the probability of its belonging to each class must be calculated based on word frequencies.

Maximization (M-step) — Updating parameters. Texts are reassigned to the group to which they are most likely to belong (based on calculated probabilities). For example, the text "Every phone call scares me" remains in class A because the words 'scares' and "call" are characteristic of stressful discourse. After that, the frequency distribution of lexemes in each class is updated to reflect the new distribution.

Repeat steps E and M. You need to repeat steps E and M several times until the texts stop changing groups. After several iterations, the model finds that:

Class A (stressful texts): 1, 3, 5

Class B (non-stressful texts): 2, 4, 6

The EM method allows us to find hidden text classes, even if we do not know the correct labels at first. Using the probabilities of words in each class, the algorithm gradually classifies the texts correctly. This approach is more appropriate for text clustering when we do not have predefined classes.

Support vector machines or SVM. In general, the support vector method is considered a classification approach, but it can be used in both classification and regression tasks. It can easily handle multiple continuous and categorical variables. SVM creates a hyperplane in a multidimensional space to separate different classes. SVM generates the optimal hyperplane iteratively, which is used to minimize error. The main idea of SVM is to find the maximum margin hyperplane (MMH) that best separates the data set into classes.

Support vectors are data points that are closest to the hyperplane. These points will better determine the dividing line by calculating the margins. These points are more relevant to the construction of the classifier.

A hyperplane is a decision-making plane that separates a set of objects that have different class affiliations.

A margin is the gap between two lines at the closest points of a class. The margin is calculated as the perpendicular distance from the line to the support vectors or closest points. If the margin between classes is larger, it is considered a good margin; a smaller

margin is considered bad.

The main goal is to separate the given data set in the best possible way. The distance between the nearest points is called the margin. The goal is to select a hyperplane with the maximum possible margin between the support vectors in a given data set. SVM searches for the maximum margin hyperplane in the following steps:

Step 1. Create hyperplanes that best separate the classes. The graph on the left shows three hyperplanes, black, blue, and orange. Here, blue and orange have higher classification errors, but black correctly separates the two classes.

Step 2. Select the right hyperplane with the maximum segregation from any nearest data point, as shown in Figure 23.

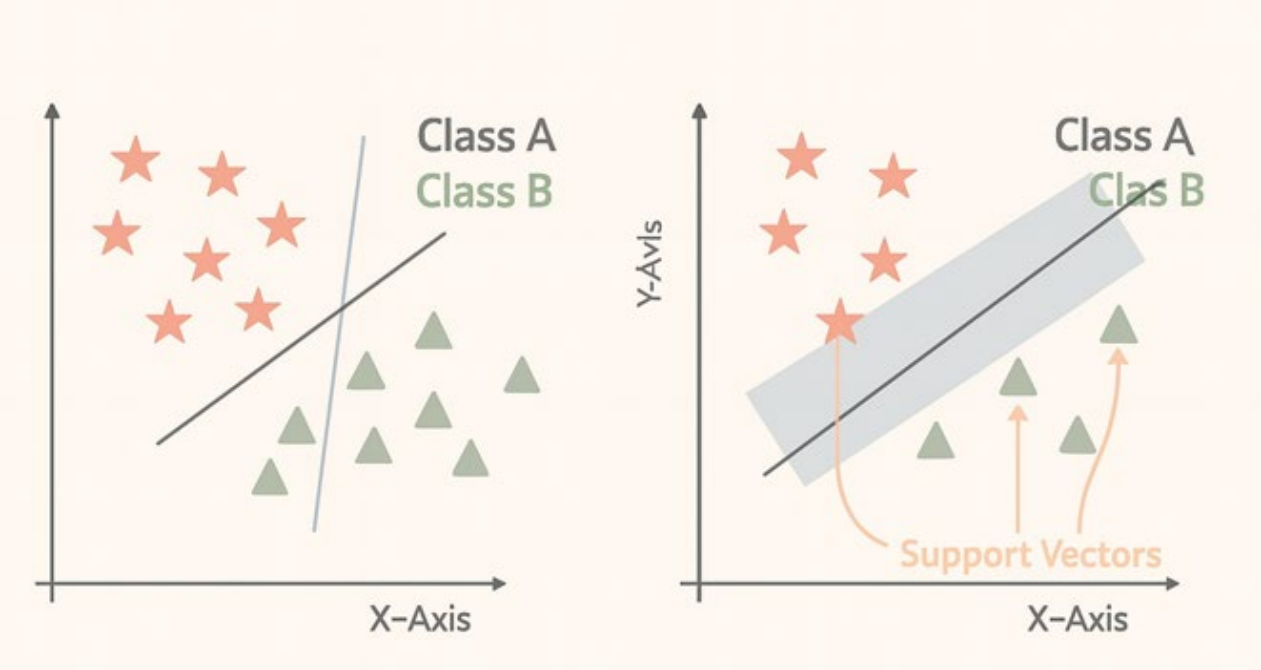


Figure 23 - Support vector machines in classification

Some problems cannot be solved using a linear hyperplane, as shown in the figure below (Figure 24).

In such a situation, SVM uses a kernel to transform the input space into a more measurable space, as shown on the right. Data points are plotted on the x-axis and z-axis (Z is the square of the sum of x and y: $z = x^2 = y^2$). Now these points can be easily separated using linear separation..

Figure 24 - Working with nonlinear and discontinuous planes

Logistic regression is one of the simplest and most effective classification algorithms. It is used to predict the probability of an object belonging to a certain class.

Logistic regression works using a sigmoid activation function, which converts input data into values from 0 to 1. If the result is above a certain threshold (for example, 0.5), the model assigns the object to one class, otherwise to another.

Below is the algorithm for logistic regression for text classification

1. Text preparation. The text is converted into a numerical format, such as Bag-of-Words or TF-IDF.

2. Model training. The input features (vectorized text) are passed to logistic regression. The model learns to determine the weights for each word.

3. Calculation of the probability of belonging to a class. A linear combination of weights and inputs is calculated (17):

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b \quad (17)$$

The sigmoid function is used to convert the z value into probability:

$$P(class = 1) = \frac{1}{1+e^{-z}} \quad (18)$$

If P > 0.5, the text belongs to class 1, otherwise to class 0.

1. Model evaluation. Accuracy is verified on test data.

First of all, one of the strengths of logistic regression is its ease of implementation, which allows for rapid model training even on limited computing resources. They are also notable for their flexibility — in cases where the amount of training data is sufficient, these methods demonstrate stable performance for a wide range of text classification tasks. In addition, one of the key aspects of their practical application is the interpretability of the results. Analysis of the weight coefficients for features allows us to identify which words have the greatest influence on the decision made, which is important in terms of model explainability.

At the same time, such approaches have a number of limitations. In particular, their effectiveness is significantly reduced in cases of complex or contextually rich tasks where deeper semantic relationships need to be modeled. In such situations, deep neural architectures, in particular models based on transformers or recurrent networks, prove to be more suitable. In addition, the effectiveness of simple models depends significantly on the quality of input feature preparation. Preliminary linguistic processing of the text, including lemmatization, noise removal, and vectorization, is a critically important step in achieving adequate performance.

Decision trees. Text classification involves assigning predefined categories or labels to text documents based on their content. Decision trees are hierarchical tree structures that recursively divide the feature space based on the values of the input features. They are particularly well suited for classification tasks due to their simplicity, interpretability, and ability to handle nonlinear relationships.

Decision trees provide a clear and understandable model for text classification, making them an excellent choice for tasks where interpretability is as important as predictive power. However, their inherent simplicity can lead to problems when working with very complex or nuanced text data, prompting practitioners to explore more sophisticated or complex improvement methods.

Decision trees are a machine learning algorithm that breaks data into subgroups based on conditions to obtain the best solution for each class. They work by creating a tree where each node of the tree corresponds to a specific question (condition) based

on features, and each leaf of the tree contains a class prediction. For text classification, decision trees can be used in conjunction with methods for converting text into numerical features.

The algorithm for text classification based on decision trees is shown below.

1. Data preparation. Convert texts into numerical format using Bag of Words or TF-IDF so that the decision tree can work with the data.

2. Build a decision tree for text classification. Train the model using the decision tree algorithm.

The algorithm analyzes which words best distinguish between text classes. It creates nodes where each question determines whether the text belongs to a particular class.

The decision tree consists of a root node — the starting point containing the entire dataset, nodes — decision points based on a specific feature, branches — paths between nodes showing the direction of division, and leaf nodes — final decisions that determine the class.

The process of building a decision tree begins with the selection of a feature that provides the most effective separation of classes. For this, formal criteria such as information gain, entropy, or the Gini index are used. The selected feature becomes the root node of the tree.

Next, the training corpus is recursively divided into subsets according to the selected features. Each node of the tree is responsible for a new division, which is performed based on the most informative word or group of words, allowing for improved homogeneity of classification labels in subsets. This division continues until the stopping conditions are met: for example, when all texts in the subset belong to the same class or when further branching does not lead to a statistically significant improvement in the accuracy of the model.

Once the tree has been constructed, the model is ready to classify new texts. The decision-making process is carried out by traversing the tree from top to bottom. At each node, the text is checked for the presence of the term that was used as a dividing feature at the corresponding level. Depending on the result of the check, the appropriate

branch of the tree is selected, along which the analysis proceeds. When a leaf node is reached, the text is assigned the appropriate class label.

Decision trees are an effective tool for basic text classification tasks due to their transparency, interpretability, and ability to reveal structural dependencies between lexical features of the text and their belonging to classification categories.

Recurrent neural networks (RNN) [55] and their modifications. Recurrent neural networks (RNN) play an important role in modern systems for processing sequential information, in particular natural language. Unlike classical neural networks, RNNs have recurrent connections that allow them to accumulate information from previous steps of text processing, i.e., to take context into account when analyzing each new element of the sequence.

However, basic RNN architectures have a significant limitation — the problem of gradient vanishing or exploding, which complicates the training of models with long dependencies between text elements. This shortcoming has been overcome in modified RNN structures, particularly in long short-term memory (LSTM) models and gated recurrent units (GRU) [18].

The LSTM (Long Short-Term Memory) [90] architecture was proposed as a solution for preserving important contextual dependencies in text. Using special control mechanisms (input, output, and forget gates), the model can selectively store or ignore information from previous steps. Thanks to this property, LSTM is widely used in tasks such as text sentiment analysis, topic classification, text generation, and automatic moderation filtering.

Another option is GRU (Gated Recurrent Unit), an architecture that simplifies the LSTM [74] structure by reducing the number of parameters without significant loss of accuracy. GRU is particularly useful in cases of limited training data or when computational costs need to be reduced. It provides comparable quality of text dependency modeling, making it suitable for implementing compact emotion classification or stress detection systems in texts.

Systems based on RNN, LSTM, and GRU [19] are effectively used in a number of natural language processing tasks: tone classification, emotional coloring detection,

automatic referencing, translation, and dialogue agent construction. In the context of analyzing emotional markers in Ukrainian, the use of such models makes it possible to take into account sequential syntactic structures and context, which is especially important when processing complex syntactic constructions and stylistic figures.

At the current stage of development of natural language processing (NLP), transformers have become the main architectural paradigm, providing high efficiency in a wide range of linguistic tasks, in particular in text classification tasks. These models are based on a self-attention mechanism that allows the model to adaptively evaluate the significance of each element of a text sequence relative to others, regardless of their position in the context. Thanks to this, transformers demonstrate excellent results when processing complex syntactic structures and long texts.

The classic text processing cycle using a transformer model involves several consecutive stages. The first stage involves tokenizing the input text — breaking it down into lexemes or sublexemes using specialized tokenizers such as WordPiece (in the BERT architecture) or Byte-Pair Encoding (in GPT models). The next step is token vectorization: each unit receives a numerical representation in the form of a vector, which is formed on the basis of pre-trained embeddings and supplemented with information about the positional structure (position embeddings).

The key mechanism of transformers is self-attention [79], in which each input element is compared with all others in the sequence to determine which ones are most important for the current processing. This approach allows the model to identify both local and global dependencies in the text. The data then passes through multi-head attention and normalization blocks [85], allowing the model to generalize the context at several levels of abstraction. The final stage is classification based on output vectors, usually implemented through a dense neural layer with a softmax or sigmoid function, depending on the type of task.

Among the most representative transformer models used in text classification, BERT, GPT, RoBERTa, XLNet, and DistilBERT are worth highlighting.

BERT (Bidirectional Encoder Representations from Transformers) is a model with a bidirectional context encoding mechanism that allows both preceding and

following words to be taken into account simultaneously. Thanks to this, BERT demonstrates high accuracy in tone analysis, query classification, and automatic text understanding tasks. The model is actively used in Google's search algorithms.

GPT (Generative Pre-trained Transformer) is an autoregressive model focused on text generation, which can also be adapted for classification through retraining. Context in GPT is processed from left to right, which distinguishes it from BERT.

RoBERTa (Robustly Optimized BERT Pretraining Approach) is an improvement on the basic BERT architecture, based on eliminating the task of masking the next sentence and increasing the size of the training corpus. This allows the model to achieve better results in a number of NLP tasks, including classification.

XLNet is a model that combines the advantages of BERT and GPT, implementing a generative approach to modeling sequences with shuffled token orders. This allows for flexibility when working with long contexts.

DistilBERT is a compromise version of BERT with fewer parameters and faster inference. It is used in cases of limited resources with a slight decrease in accuracy.

Transformer architectures have a number of advantages. In particular, they provide deep contextual understanding of texts, support multifunctionality (translation, generation, tone analysis, etc.), demonstrate high accuracy in large-scale training, and allow fine-tuning on specialized corpora to adapt to specific tasks.

The use of transformers in the context of emotional state analysis, particularly stress, is extremely promising due to their ability to take into account complex semantic-syntactic structures, hidden associations between words, and respond flexibly to the ambiguity of linguistic expressions.

4.2. Clustering of emotional patterns

The clustering task is similar to the classification task, it is its logical extension, but the difference is that the classes of the data set under study are not predefined. Synonyms for the term “clustering” are “automatic classification”, “unsupervised learning”, and “taxonomy”.

Clustering is designed to divide a set of objects into homogeneous groups of clusters or classes. If the sample data are represented as points in the feature space, then the clustering task is reduced to identifying "clusters of points". The purpose of clustering is to find existing structures. Clustering is a descriptive procedure, it does not make any statistical conclusions, but it makes it possible to conduct an exploratory analysis and study the "data structure".

The very concept of "cluster" is ambiguously defined. The term cluster is translated as "cluster" or "bunch". A cluster can be characterized as a group of objects that have common properties. Figure 25 shows a comparison of classification and clustering.

Two characteristics of a cluster are internal homogeneity and external isolation.

The question that analysts ask themselves when solving many problems is how to organize data into clear structures, i.e., how to develop taxonomies.

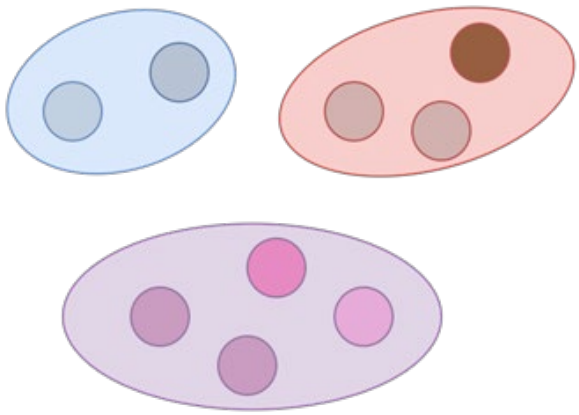


Figure 25 – Visualization of clustering

Table 19 compares classification and clustering, and Figure 26 shows a visualization.

Clusters can be non-overlapping, exclusive, or overlapping (Figure 27).

It should be noted that different methods of cluster analysis can result in clusters of different shapes. For example, there may be "chain" type clusters, where clusters are represented by long "chains", elongated clusters, etc., and some methods can create clusters of arbitrary shapes.

Table 19 – Comparison of classification and clustering methods

Characteristics	Classification	Clustering
Controllability of learning	Controlled learning	Uncontrolled learning
Strategy	Learning with a teache (Supervised)	Learning without a teacher (Unsupervised)
Class mark presence	The training set is accompanied by a label indicating the class to which the observation belongs	Tags for the training set are unknown
Basis for classification	New data is classified based on the training set	A set of data is given with the aim of establishing the existence of classes or clusters of data

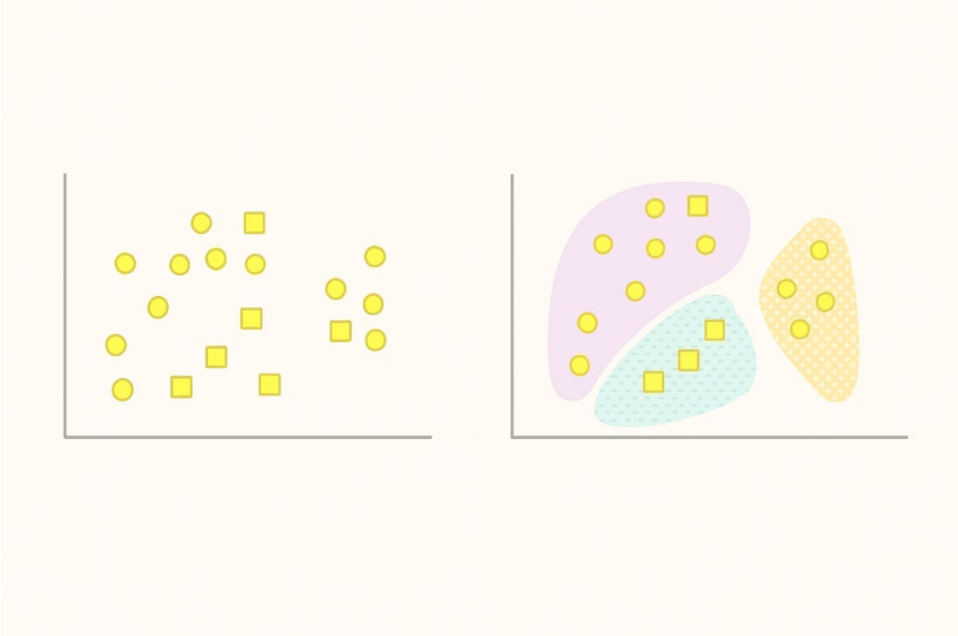


Figure 26 – Visualization of classification and clustering processes

Different methods may aim to create clusters of certain sizes (e.g., small or large) or assume the presence of clusters of different sizes in the dataset. Some cluster analysis methods are particularly sensitive to noise or outliers, while others are less so.

As a result of applying different clustering methods, different results may be obtained, which is normal and is a feature of the operation of a particular algorithm.

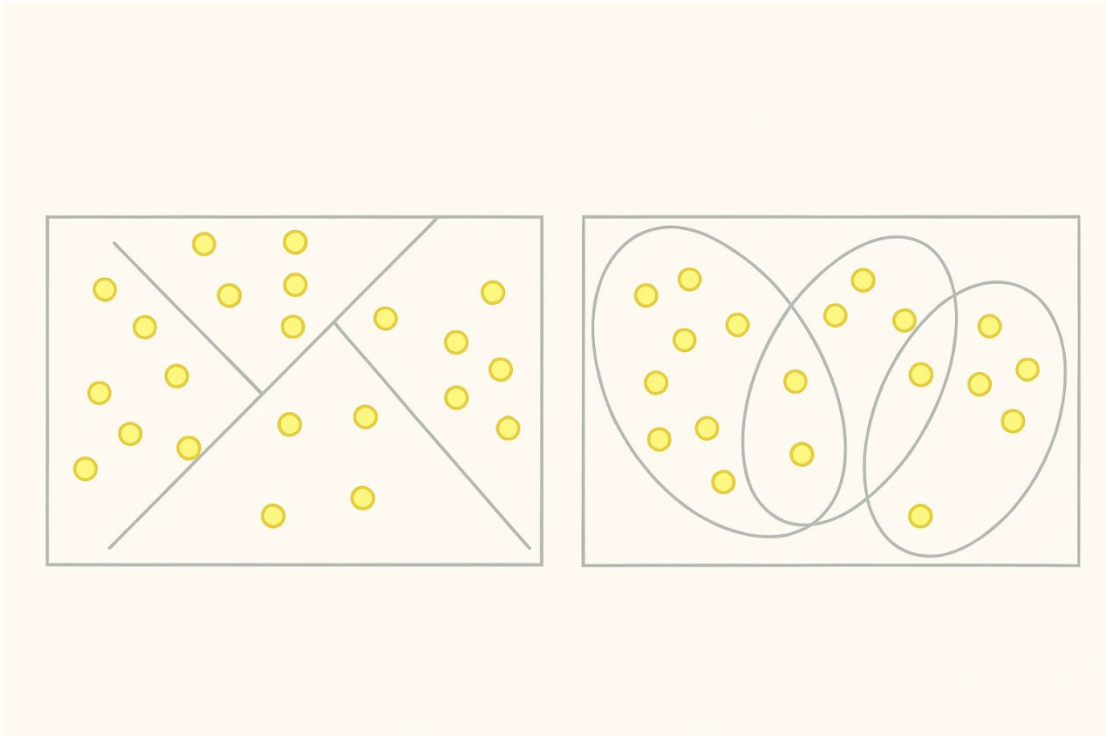


Figure 27 - Displaying non-overlapping and overlapping clusters

In hard clustering, each object belongs to exactly one cluster. In soft clustering, an object can belong to one or more clusters. Belonging can be partial, i.e., objects can belong to certain clusters more than to others (Figure 28).

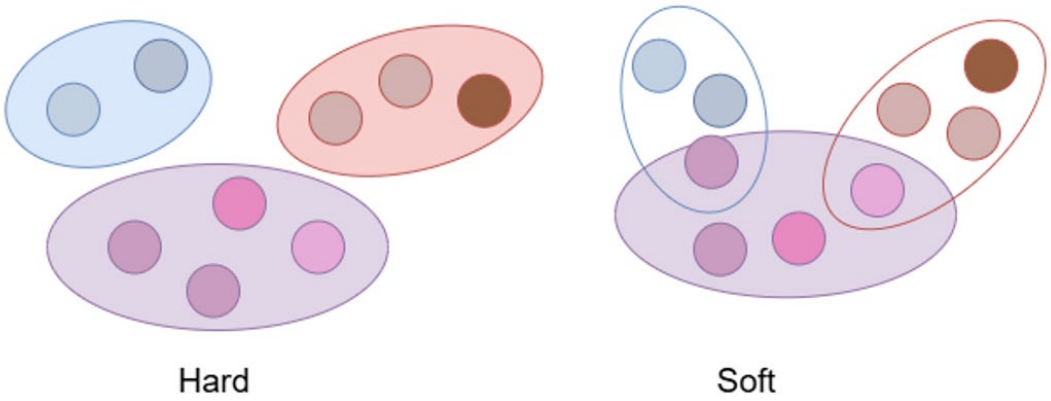


Figure 28 – Visualization of hard and soft clustering

In hierarchical clustering (Figure 29), clusters are iteratively merged in a hierarchical manner, ultimately ending up with a single root or supercluster. You can also think of hierarchical clustering as a binary tree. All clustering methods that do not

follow this principle can simply be described as flat clustering, but they are sometimes also referred to as non-hierarchical or partitioning. You can always convert hierarchical clustering to flat clustering by "cutting" the tree horizontally at a level of your choice.

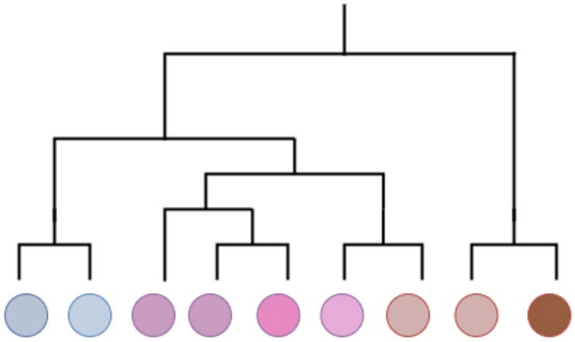


Figure 29 - Visualization of hierarchical clustering

Hierarchical methods can be further divided into two subcategories. Bottom-up agglomeration methods start by placing each object in a separate cluster and then continue to merge them. Top-down splitting methods do the opposite: they start with the root and continue to split it until only individual objects remain.

These features should be taken into account when choosing a clustering method. More than a hundred different clustering algorithms have been developed so far.

Text clustering can be performed at the document, sentence, or word level. Document-level clustering is used to regroup documents on the same topic. Document clustering is used in news articles, emails, search engines, and more. Sentence-level clustering is used to group sentences from different documents. An example is tweet analysis. For word-level clustering, word clusters are groups of words based on a common theme. The easiest way to build a cluster is by collecting synonyms for a particular word. For example, WordNet is a lexical database for the English language that groups English words into sets of synonyms called synsets.

In text clustering, it is important to measure the similarity of words. Words can be similar lexically or semantically. With lexical similarity, words are lexically similar if they have a similar sequence of characters. Lexical similarity can be measured using string-based algorithms that work with sequences of strings and comp.

In text clustering, it is important to measure the similarity of words. Words can be lexically or semantically similar. With lexical similarity, words are lexically similar if they have a similar character sequence. Lexical similarity can be measured using string-based algorithms that work with string sequences and character composition. For semantic similarity, words are semantically similar if they have the same meaning, are opposites of each other, are used in the same way, are used in the same context, or one is a type of the other. Semantic similarity can be measured using corpus-based or knowledge-based algorithms.

Some of the metrics for calculating the similarity between two pieces of text are Jaccard coefficient, cosine similarity, Euclidean distance, etc.

The process of clustering text documents involves the implementation of a number of sequential steps, each of which is crucial for achieving a correct and interpretable cluster structure. Despite the fact that in practice the clustering process is often accompanied by difficulties, trial and error, and hyperparameter settings, it can be represented in general as a sequential chain of operations performed on a text corpus.

The first stage is text data pre-processing, which aims to reduce the influence of noise components, such as stop words, morphological variants, non-informative units, and normalize the lexical structure. The main procedures at this stage include lemmata.

The first stage is text data pre-processing, which aims to reduce the influence of noise components, such as stop words, morphological variants, non-informative units, and normalize the lexical composition. The main procedures at this stage include lemmatization or stemming, removal of punctuation marks, lowercase words, and cleaning the text from functionally insignificant lexemes. The use of such techniques can significantly reduce the sparsity of vector representations and reduce the dimensionality of the feature space.

The next critical step is feature extraction, which is the transformation of unstructured text into a formalized numerical representation suitable for algorithmic processing. The most common approaches at this stage are the use of frequency-based models, such as Bag-of-Words, TF, TF-IDF, or more modern contextualized vectorizations, such as Word2Vec, FastText, BERT embeddings. These vector

representations capture the frequency, semantic, and syntactic properties of lexical items, allowing them to be used effectively in grouping tasks.

The third stage is clustering itself, which involves grouping text objects based on the similarity of their feature vectors. Depending on the nature of the data, the number of observations, and the goal, algorithms such as K-Means, hierarchical clustering (agglomerative or divisive), DBSCAN, HDBSCAN, Spectral Clustering, etc. can be used for this purpose. The key task is to ensure theThe third stage is clustering itself, which involves grouping text objects based on the similarity of their feature vectors. Depending on the nature of the data, the number of observations, and the goal, algorithms such as K-Means, hierarchical clustering (agglomerative or divisive), DBSCAN, HDBSCAN, Spectral Clustering, etc. can be used for this purpose. The key task is to ensure that the objects in the same cluster have similar thematic or stylistic properties.

Thus, in a simplified, idealized form, the process of text clustering can be represented as a sequence of the following steps: 1) normalization of text data; 2) extraction of relevant features; 3) application of the clustering algorithm followed by analysis of the resulting groups. It is worth noting that in real-world conditions, these stages can be refined or repeated based on empirical results, as well as depending on the specifics of the linguistic material or application task. Figure 30 shows an augmented clustering algorithm that takes into account the peculiarities of working with text data.

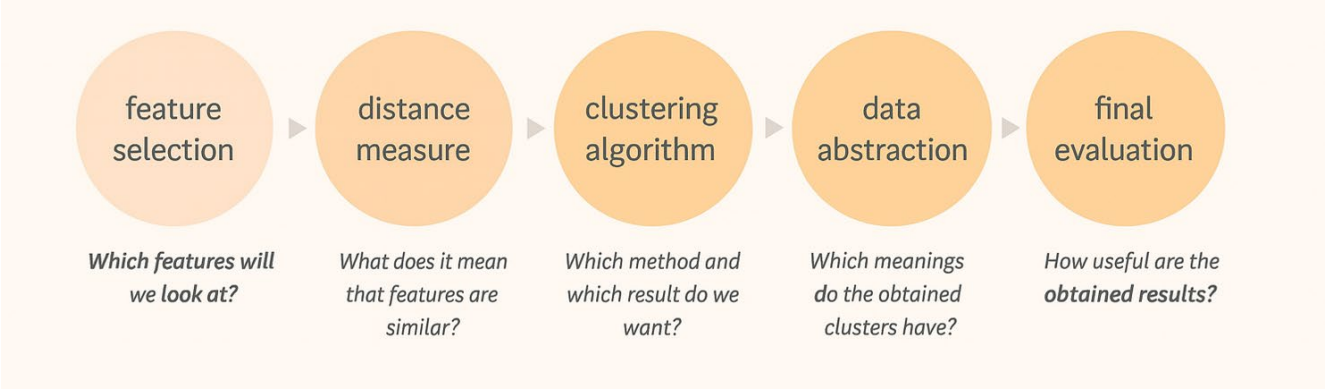


Figure 30 - The sequence of text clustering

Below is a description of the main approaches to text clustering.

The k-nearest neighbors method is a simple non-parametric classification method where distances, usually Euclidean, calculated to all other objects are used to classify objects within a property space. The objects with the smallest distance are selected and allocated to a separate class.

The k-nearest neighbors method is a metric algorithm for automatic object classification. The basic principle of the k-nearest neighbor method is that an object is assigned to the class that is most common among the neighbors of a given element. Neighbors are taken from the set of objects whose classes are already known, and based on the key value k for this method, it is calculated which class is the most numerous among them. Each object has a finite number of attributes (dimensions). It is assumed that there is a certain set of objects with an existing classification.

The k-Nearest Neighbor (k-NN) method is a simple classification method based on comparing the similarity of a new sample with already known knowledge samples. To classify a new sample, k-NN compares it with several closest knowledge samples belonging to different classes and determines which class has the largest number of close samples.

In text classification, each text can be represented as a vector of words, for example, using TF-IDF. To compare the similarity of texts, various metrics can be used, such as cosine similarity. Further, to classify a new text, k-NN compares it with the k closest texts with already known classes and determines which class most of them belong to.

One of the advantages of the k-NN method is its simplicity and ability to work with high-dimensional data, including text data. In addition, k-NN does not require preprocessing and model training, which makes it fast and efficient. However, k-NN may not be as accurate as more sophisticated classification methods and may require a large amount of memory to store already known knowledge samples.

Hierarchy Algorithms. Hierarchical clustering belongs to the class of unsupervised machine learning methods that allow you to identify nested structure in data by gradually combining or separating objects. It results in the construction of a

dendrogram, a cluster tree that visually displays the hierarchical relationships between documents. The method starts with treating each data element (for example, a text document) as a separate cluster. The algorithm then gradually merges the closest pairs of clusters based on the selected metric until a single cluster is formed that covers all objects.

The main linking strategies are Single linkage - combining clusters by the minimum distance between their elements; Complete linkage - using the maximum inter-cluster distance; Average linkage - clustering based on the average distance between all pairs of elements.

The advantage of the hierarchical approach is that there is no need to pre-determine the number of clusters, which makes it convenient for studying an unknown data structure. In addition, the dendrogram as a visualization tool ensures the interpretability of the results. However, the method has scalability limitations, which makes it difficult to apply to large corpora of text, and can be sensitive to noisy data.

Methods based density-based methods

One of the leading clustering methods in cases of complex, irregular data structure is the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm. Its principle of operation is based on the idea of the spatial density of the distribution of points in a multidimensional feature space. Unlike classical methods, such as K-means clustering or agglomerative hierarchical clustering, DBSCAN does not make a priori assumptions about the shape of clusters (e.g., spherical or compact) and provides effective detection of clusters of arbitrary configuration.Formally, DBSCAN defines clusters as areas of high object density separated by areas of low density, which allows it not only to identify structured groups but also to label outliers - points that do not belong to any cluster. An important advantage of this method is its ability to handle clusters of any shape, including elongated, curved, or fractal structures, as well as automatic noise detection in the input data.

The functioning of the algorithm is controlled by two key parameters: the neighborhood radius ε (epsilon) and the minimum number of points MinPts required to form a cluster. The parameter ε determines the maximum distance at which objects

are considered neighboring. A value of ε that is too small results in a large number of noise points, while a value that is too large results in the aggregation of several clusters into one. Determining the optimal value of ε is usually done empirically or by analyzing distance graphs (e.g., the k-distance method).

The MinPts parameter is usually set by the rule: MinPts ≥ D + 1, where D is the dimensionality of the feature space. In most application scenarios, the recommended value of MinPts is at least three.

According to the conceptual model, the algorithm classifies all sample points into three categories (Figure 31):

1. Core points - have at least MinPts neighbors within the ε-neighborhood.
2. Boundary points - have less than MinPts neighbors, but fall into the ε-neighborhood of the core point.
3. Noise points - points that do not belong to any cluster and do not have a sufficient number of close neighbors.

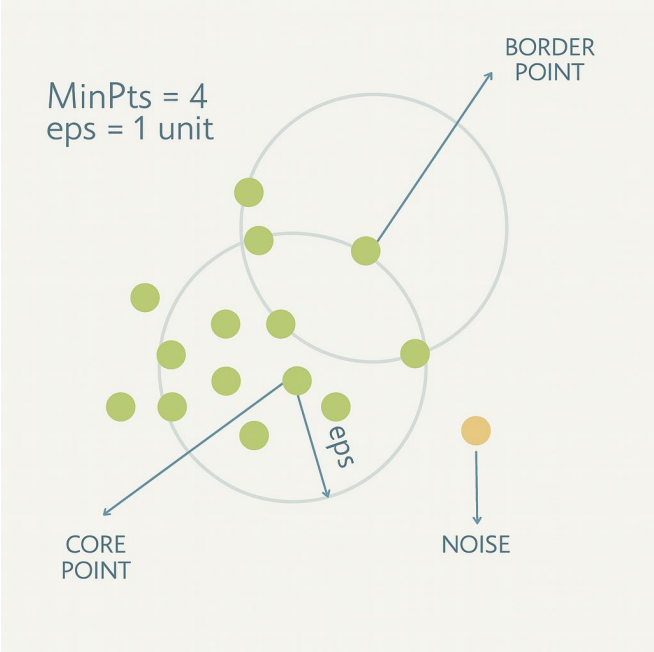


Figure 31 - Visualization of the dbscan algorithm with MinPts and eps units

The algorithm is performed in several stages. First, the number of neighbors within the ε-neighborhood is calculated for each point. If there are enough neighbors, this point becomes the main point. Then, for each unanalyzed core point, a new cluster is created, which recursively joins all points that are reachable by density. This

reachability is determined by constructing a chain of points connected by consecutive ε-neighborhoods, provided that there is at least one basic point in the chain. After this process is completed, all remaining points are labeled as noise. Thus, DBSCAN demonstrates high robustness to outliers and provides effective clustering in situations where classical Euclidean geometry is not relevant to describe the structural properties of the data. Its application is appropriate for text analysis, anomaly detection, geospatial data processing, and research with complex semantic topology.

Among the modern modifications of hierarchical methods, an important place is occupied by HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise), which integrates hierarchical clustering approaches with density-based clustering. Compared to the classical DBSCAN algorithm, HDBSCAN does not require a predefined number of clusters. Instead, it automatically detects the optimal number of groups based on the local density of the objects. Another significant advantage is the ability to estimate the degree to which an object belongs to a particular cluster, which is especially important when analyzing heterogeneous and complexly structured text data. To implement this method, you need to use the corresponding hdbscan library, which provides high computational efficiency and flexibility in application.

Grid-based methods use a grid data structure with multiple resolutions. It quantizes object areas into a finite number of cells that form the grid structure on which all clustering operations are implemented. The advantage of the method is fast processing time, which is usually independent of the number of data objects and depends only on a few cells in each dimension of the quantized space.

An example of a grid-based approach is STING, which examines statistical data stored in grid cells, WaveCluster, which clusters objects using a wavelet transform, and CLIQUE, which defines a grid and density-based approach for clustering in high-dimensional space.

Grid-based methods use a grid data structure with multiple resolutions. It quantizes object areas into a finite number of cells that form a grid structure on which all clustering operations are implemented. The advantage of the method is fast

processing time, which is usually independent of the number of data objects and depends only on a few cells in each dimension of the quantized space.

An example of a grid-based approach is STING, which examines statistical data stored in grid cells, WaveCluster, which clusters objects using a wavelet transform, and CLIQUE, which defines a grid and density-based approach for clustering in high-dimensional data space.

STING is a multi-resolution grid-based clustering method in which the spatial domain is divided into rectangular cells. Usually there are several levels of such rectangular cells corresponding to several levels of resolution, and these cells form a hierarchical mechanism in which each cell at a high level is divided to form several cells at the next lower level. Statistical data about the attributes in each grid cell (including the average, maximum, and minimum values) are pre-calculated and stored.

The statistical parameters of higher-level cells can be simply calculated from the parameters of lower-level cells. These parameters include the following: an attribute-independent parameter, count, and attribute-dependent parameters, mean.

The statistical parameters of higher-level cells can be simply calculated from the parameters of lower-level cells. These parameters include the following: an attribute-independent parameter, count, and attribute-dependent parameters, mean, stdev, min, max, and the type of distribution that the attribute value in the cell follows, including normal, uniform, exponential, or missing.

When the records are loaded into the database, the count, mean, stdev, min, and a max parameters of the lower-level cells are calculated directly from the records. The distribution value can be assigned by the user if the distribution type is known in advance, or it can be obtained by hypothesis testing, including the $\chi 2$ test.

The type of distribution of higher-level cells that can be calculated depends on most of the distribution types of the corresponding lower-level cells in combination with the threshold filtering procedure. If the lower-level cell distributions are not consistent with each other and do not pass the threshold test, the high-level cell distribution type is set to none.

Statistical parameters can be used in top-down grid-based approaches as follows.

First, you determine the level in the hierarchical architecture from which the query response procedure should start. This level usually contains a small number of cells. For each cell in the current level, a confidence interval (or estimated probability range) can be calculated, which reflects the relevance of the cell to the given query.

WaveCluster is a multi-resolution clustering algorithm that first summarizes records by overlaying a multidimensional grid architecture on the data space. It can use a wavelet transform to change the original feature space, finding dense domains in the transformed space. In this method, each grid cell summarizes the data of a group of points.

In this method, each grid cell summarizes the data of a group of points that are displayed in the cell. This summed data is usually placed in main memory for use by a multi-resolution wavelet transform and subsequent cluster analysis.

Wavelet transform is a signal processing approach that decomposes a signal into multiple frequency sub-bands. A wavelet model can be used for d-dimensional signals by performing a one-dimensional wavelet transform d times. When applying the wavelet transform, the data is modified to preserve the relative distance between objects at multiple levels of resolution. This makes natural clusters in the data more visible. Clusters can be recognized by looking for dense areas in the new domain.

The advantage of the wavelet transform is that it provides a teacherless clustering: it requires hat-shaped filters that emphasize areas where points cluster while suppressing weaker data outside the cluster.

CLIQUE is a density and grid-based subspace clustering algorithm. So first, let's look at what a grid and density-based clustering method is.

The CLIQUE algorithm uses a density and grid-based method, that is, a subspace clustering algorithm, and finds a cluster by taking the density threshold and the number of grids as input parameters. It is specially designed to handle datasets with a large number of dimensions. The CLIQUE algorithm is highly scalable with respect to the value of records and the number of dimensions in the dataset because it is grid-based and effectively utilizes the a priori property.

The a priori approach states that if an X-dimensional unit is dense, then all of its

projections in X-1-dimensional space are also dense.

This means that dense regions in a given subspace should create dense regions when projected onto a low-dimensional subspace. CLIQUE restricts the search for high-dimensional dense cells to the intersection of dense cells in the subspace because CLIQUE uses a priori properties.

The CLIQUE algorithm first divides the data space into grids. It does this by dividing each dimension into equal intervals called units. After that, it identifies dense units. A unit is dense if the data points in it exceed. The CLIQUE algorithm first divides the data space into grids. It does this by dividing each dimension into equal intervals called units. After that, it identifies dense units. A unit is dense if its data points exceed a threshold value.

After the algorithm finds dense units along one dimension, the algorithm tries to find dense units along two dimensions and works until all dense units along the entire dimension are found

Після After finding all dense cells in all dimensions, the algorithm proceeds to find the largest set ("cluster") of related dense cells. Finally, the CLIQUE algorithm generates a minimal cluster description. Clusters are then generated from all dense subspaces using an a priori approach.

The advantages of CLIQUE are that it is a subspace clustering algorithm that outperforms K-means, DBSCAN, and Farthest First in both runtime and accuracy. CLIQUE can find clusters of any shape and is capable of finding any number of clusters in any number of dimensions where the number is not defined by a parameter, and is one of the simplest methods and interpretability of results.

The main disadvantage of the CLIQUE algorithm is that if the cell size is not suitable for a set of very high values, then too much evaluation will be performed and the correct cluster will not be found.

Agglomerative clustering is one of the most common types of hierarchical clustering, which is implemented in accordance with the bottom-up principle, i.e., the bottom-up strategy. This approach involves initially treating each object as a separate cluster, followed by iterative merging of the closest clusters based on a certain distance

metric. Depending on the chosen merging criterion (single link, complete link, average link, etc.), a dendrogram is formed - a tree of clusters that allows you to explore the structure of relationships between objects. The agglomerative clustering algorithm is actively supported in modern software libraries, in particular in scikit-learn, which ensures its availability for a wide range of applications in the field of text corpus analysis.

Another promising approach to text clustering is spectral clustering, which is based on the preliminary formation of a graph of adjacency between objects, in particular, based on cosine similarity or Gaussian kernel. The constructed graph is characterized by spectral properties that are determined by analyzing the eigenvectors of a normalized or non-normalized Laplace matrix.

The use of spectral decomposition allows you to project multidimensional data into a lower dimensional space, in which further clustering is performed, usually using the K-means algorithm. This method is particularly effective when the data has a complex, nonlinear structure or cannot be divided in a simple Euclidean way.

In general, the combination of hierarchical, spectral, and density approaches to clustering provides an extension of the researcher's analytical toolkit and allows for the adaptation of classification procedures to the peculiarities of real language material.

In general, the choice of a clustering technique in the task of analyzing text corpora should take into account the amount of data, the expected complexity of the structures, and the need for interpretability of the resultsThe use of spectral decomposition allows you to project multidimensional data into a lower dimensional space, in which further clustering is performed, usually using the K-means algorithm. This method is particularly effective when the data has a complex, nonlinear structure or cannot be divided in a simple Euclidean way.

In general, the combination of hierarchical, spectral, and density approaches to clustering provides an extension of the researcher's analytical toolkit and allows for the adaptation of classification procedures to the peculiarities of real language material.

In general, the choice of a clustering technique in the task of analyzing text corpora should take into account the amount of data, the expected complexity of the

structures, and the need for interpretability of the results.

Evaluation of clustering quality - The quality of clustering plays a key role in the process of analyzing textual data, as it determines the accuracy of detecting latent structure in the corpus. The appropriate evaluation allows not only comparing different clustering algorithms but also provides a basis for further interpretation of the obtained clusters. Evaluation methods are classified into internal and external quality measures.

Internal clustering metrics

Internal measures are based solely on the structure of the clusters obtained as a result of modeling. The main aspects are compactness (the degree to which objects are clustered within a single cluster) and separation (the distance between clusters). These characteristics are formalized as intra-cluster variance and inter-cluster distance, respectively.

Typical methods of internal evaluation include the Silhouette Score and the Davis-Boulden Index.

The Silhouette Score is an index that measures the degree to which each object belongs to a particular cluster by comparing its similarity to the nearest neighboring cluster. The value of this index ranges from -1 to 1, where 1 corresponds to a perfect classification, 0 to a borderline case, and negative values to incorrect cluster inclusion.

The Davies-Bouldin Index is a metric that reflects the average similarity between each cluster and the most similar one. The value of the index is a function of the ratio between the internal variance of the clusters and the distance to the nearest neighbor. A smaller index value indicates better cluster separation. One of the advantages of this approach is its invariance to the shape of the clusters and the ability to work with any number of clusters.

In addition, to assess the stability of clustering, approaches with variation of model parameters are used, for example, adding variables or comparing clustering results obtained by different algorithms, for example, k-means, DBSCAN, HDBSCAN.

External clustering metrics - in cases where a reference classification (ground-truth) is available, it is possible to use external metrics that allow comparing the

predicted clusters with the actual labels.

The **Adjusted Rand Index** (ARI) is a statistical indicator that measures the similarity between two data distributions. The ARI value is normalized in such a way that 1 corresponds to a complete match, and a value close to 0 corresponds to a random distribution. If the value is negative, we can talk about a contradictory classification.

Normalized Mutual Information (NMI) is an indicator that evaluates the mutual information between true labels and cluster assignments. It is normalized to the range [0, 1], where 1 means full correlation. An important property of this metric is that it is invariant to permutation of label values and symmetric with respect to the two partitions. In cases where there is no underlying labeling, NMI can be used to reconcile the results of independent classifications.

In general, the choice of a metric for assessing the quality of clustering depends on the presence or absence of reference labels, as well as on the nature of the data distribution in a multidimensional space. A systematic combination of internal and external evaluations allows for a more informed analysis of clustering results, which is a prerequisite when working with text corpora, in particular in the context of identifying thematic or emotional structures.

Visualization of clusters

In the process of processing textual information, one of the most pressing tasks is to visualize the results of clustering. The main difficulty lies in the fact that after text vectorization (in particular, using methods such as TF-IDF), the feature space can reach hundreds or even thousands of dimensions. This high dimensionality makes it impossible to directly display the data structure in traditional two- or three-dimensional coordinates. Therefore, for effective visualization, it is necessary to use dimensionality reduction methods that allow you to preserve the most important features of space in a reduced representation.

One of the most common approaches to dimensionality reduction is Principal Component Analysis (PCA). This method is based on a linear transformation of the feature space in order to maximize the preservation of data variance in the new coordinates. PCA is characterized by high speed, which makes it attractive for

preliminary analysis of large text corpora. However, its linear nature does not allow detecting complex nonlinear structures inherent in text data.

In cases where it is important to take into account local relationships between objects, it is advisable to use the t-Distributed Stochastic Neighbor Embedding (t-SNE) method. This nonlinear algorithm projects multidimensional data into a lower dimensional space in a way that preserves the relative proximity between objects in the original space as much as possible. t-SNE is good at displaying the cluster structure of data, showing visually distinct groups. However, it should be noted.

One of the most common approaches to dimensionality reduction is Principal Component Analysis (PCA). This method is based on a linear transformation of the feature space in order to maximize the preservation of data variance in the new coordinates. PCA is characterized by high speed, which makes it attractive for preliminary analysis of large text corpora. However, its linear nature does not allow detecting complex nonlinear structures inherent in text data.

In cases where it is important to take into account local relationships between objects, it is advisable to use the t-Distributed Stochastic Neighbor Embedding (t-SNE) method. This nonlinear algorithm projects multidimensional data into a lower dimensional space in such a way as to preserve the relative proximity between objects in the original space as much as possible. t-SNE is good at displaying the cluster structure of data, showing visually distinct groups. However, it should be noted that this method does not guarantee the preservation of global distances, which may affect the interpretation of distant clusters.

Another modern approach is the Uniform Manifold Approximation and Projection (UMAP) algorithm, which combines the advantages of t-SNE with increased efficiency and the ability to preserve both local and global data structure. UMAP is based on the theory of multidimensional manifolds and uses methods of topological space modeling. Compared to t-SNE, it provides better scalability and stability of results when working with large text collections.

Thus, applying dimensionality reduction methods is a key step in visualizing clustered text data. The choice of a particular method depends on the nature of the task,

the amount of data, and the requirements for preserving structural information. PCA can be used for a quick overview, while t-SNE and UMAP are more appropriate for a deeper analysis of the cluster structure and revealing hidden relationships between text units.

4.3. Ensamble technics and methods

In the context of text classification by psycho-emotional state, where input data is usually represented as high-dimensional and sparse vectors, the LightGBM model proves to be one of the most appropriate among modern ensemble machine learning algorithms. The use of algorithmic improvements, such as Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), allows the model to provide high computational efficiency while maintaining forecasting accuracy.

In a comparative aspect, Random Forest is inferior to LightGBM due to its limited ability to model complex nonlinear dependencies, as well as insufficient efficiency when working with sparse feature matrices, which is typical for textual representations. Despite its high accuracy, XGBoost demonstrates lower performance in scaling and training time on large feature sets.

Thus, LightGBM demonstrates advantages both in terms of computational efficiency and adaptability to the features of text data. Its ability to learn accurately and quickly in conditions of high dimensionality and sparse features allows us to consider this algorithm as an optimal solution for automated classification of the psycho-emotional state of text.

The essence of this algorithm is that each tree learns from the error gradients of the previous model. LightGBM does not try all possible splits, but uses optimizations: GOSS - selects the most important examples with large errors or EFB - combines non-overlapping features to reduce the dimensionality.

The main idea is that instead of slowly training trees on all the data, LightGBM builds deeper trees faster, works efficiently with large datasets and many features, and provides high classification accuracy.

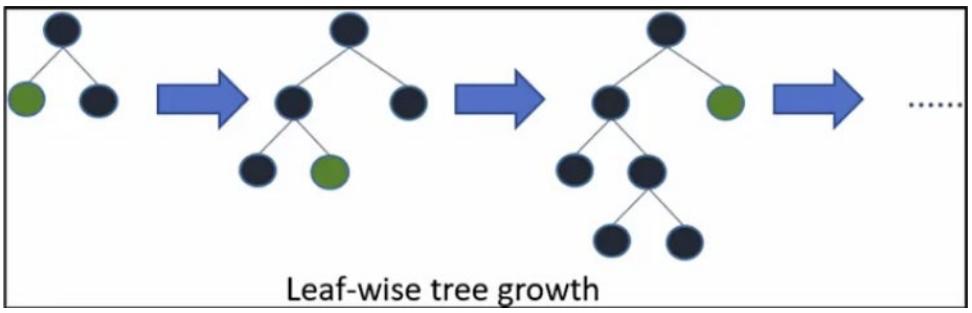


Figure 32 - Leaf-wise image in LightGBM

Figure 32 illustrates leaf-wise tree growth, a key strategy used by LightGBM as opposed to the traditional level-wise growth used in most tree algorithms.

Level-wise growth means that all nodes at the same level of the tree are split at the same time. It is balanced, but slow and less efficient, as sometimes computations are spent on "less useful" nodes. Whereas Leaf-wise growth always selects the leaf with the largest decay in the loss function and splits it. That is, the tree is built in depth, not in breadth. Each new division brings the maximum loss reduction and thus results in a more accurate model with lower losses.

Conclusion

Text classification (also known as categorization or text tagging) is a fundamental task of text mining, which consists in assigning predefined categories or labels to a text document based on its content. This process can be either manual or automated, implemented using rule sets or machine learning methods.

Clustering is a data mining task that is similar to classification, but differs in the absence of predefined categories (unsupervised learning, unsupervised learning). Its goal is to divide a set of objects (e.g., text documents) into homogeneous groups (clusters) based on their properties, without prior knowledge of existing classes.

In the context of analyzing the psycho-emotional state of individuals, classification and clustering of text data play a key role in identifying signs of stress. These methods allow you to systematize large amounts of unstructured information, such as social posts, diaries, chats, emails, or answers to open-ended questions, in order

to identify hidden emotional signals typical of stressful states.

Classification methods are used to automatically recognize the presence or absence of stress in text messages [48]. Usually, such tasks are formalized as a binary classification “stressful” / “non-stressful” or as a multi-class classification “stress”, “anxiety”, ‘depression’, “normal”. To do this, machine learning models are trained on labeled corpus [49].

Classification methods are used to automatically recognize the presence or absence of stress in text messages. Usually, such tasks are formalized as a binary classification “stressful” / “non-stressful” or as a multi-class classification “stress”, “anxiety”, ‘depression’, “normal” [56]56]. To do this, machine learning models are trained on labeled corpora of texts in which the stressful state has been previously confirmed by specialists or self-report.

This approach allows us to detect signs of psycho-emotional stress in real time and support the decisions of mental health professionals [57].

Deep learning models, such as BERT, RoBERTa, are able to consider the context and hidden semantic connections, which makes text classification especially accurate in emotionally charged situations.

Clustering allows you to detect latent groups of messages that demonstrate similar linguistic, semantic, or emotional features. Unlike classification, clustering does not require preliminary data labeling, which is a significant advantage when there are limited labels available.

Clustering is used to pre-analyze unstructured data and build a typology of messages, while classification is used to accurately determine the level or presence of stress. This hybrid approach allows for the creation of adaptive mental health support systems capable of both detecting individual changes in the user's state and analyzing collective trends.

Thus, classification and clustering form the basis for the intellectual analysis of emotionally colored text, which is crucial in early warning systems for stressful conditions, automatic mental health support, and in the field of digital psychodiagnostics in general

KAPITEL 5 / *CHAPTER 5*
SYSTEMS AND TOOLS DEVELOPMENT

5.1. Input data analysis

The open dataset "reddit-mental-illness-82" [104] is hosted on the Hugging Face platform. This dataset contains text messages from Reddit users accompanied by labels that indicate the probable psycho-emotional state of the message author.

The dataset includes more than 52 thousand text entries, structured into two primary columns: the text itself, which should be classified, and the labels to which class it was assigned by the expert. The labels' column contains eight categories of psycho-emotional conditions, in particular, obsessive-compulsive disorder (ocd - 6), bipolar disorder (bipolar - 2), attention deficit hyperactivity disorder (adhd - 0), post-traumatic stress disorder (pstd) - 7, depression (depression - 4), anxiety (anxiety - 1), borderline personality disorder (bpd - 3) and neutral (none - 5).

text string · *lengths* 108 5.63k	label class label 8 classes
How does your ADHD affect you? : Hello my fellow brethren, sisteren, and otherkin! I was diagnosed with adhd-c after almost failing out of my freshman year of college, and nearly losing my mind…	0 adhd
Talking about The Disorder too much : Does anyone else have a tendency to obsess about BPD, their symptoms, etc.? I feel like I'm constantly trying to explain myself to people and just... talking…	3 bpd
[Personal Breakthrough/WARNING: Panic Attack] West Wing's "Noël" episode - S2E10 - is maybe the greatest mainstream TV-show "coverage" of PTSD. : I've been rewatching *The West Wing* recently,…	7 ptsd
ADHD makes me feel like I'm always moving too fast while also being 10 steps behind. : For example: It always feels like my brain has 50 tabs open whenever I'm speaking to people. I start…	0 adhd
What About Bob? : For those of you who do not know the movie, What About Bob? is about a mentally ill man who is greatly helped by being told to take a vacation from his problems. It is a comedy,…	2 bipolar
Mental Illness: where the one Suffering gets less Sympathy than those around them : Don't know if this happened to anyone else who ever experienced severe mania, but even after my manic phase…	2 bipolar
Can I just say you guys are the best? : Seriously I am so glad I found this community. You guys are so supportive and whether it's advice, laughing at memes or just co-misserating you guys…	6 ocd
My uncle passed away this afternoon and I feel more remorse for not feeling anything about it than I do for his passing. : This isn't even the first time this year that I have felt... nothing for…	0 adhd
Say We're in a French Village in 1864, Who Would We Look to as "the Police"? : More specifically, would there be a unified, paid police force, or a designated constable? What would they be called…	5 none
Your depression ever get so bad that reality feels like a TV show? : I swear some nights when I'm eating dinner with the rest of my family, none of it feels real. Sometimes it's like I'm watching…	4 depression

Figure 33 - Sample from the "reddit-mental-illness-82" dataset

The website immediately divides the dataset into training, validation, and test samples. For further analysis, we will select the test sample, which contains 42113 records. Figure 33 shows that most of the records fall into the fifth, i.e., neutral psychoemotional state, and the fewest records correspond to the second state, bipolar

disorder.

The proposed emotional analysis system is based on integrating modern natural language processing libraries with a focus on the specifics of the Ukrainian language. Using such tools as Stanza and HuggingFace Transformers allows for high accuracy in recognizing emotional markers and stressful states through deep linguistic analysis, contextual vectorization, and syntactic interpretation of the text.

5.2. Features of data pre-processing

The above confusion matrix and classification report demonstrate the results of classifying psychoemotional states using logistic regression [40]. In general, the model achieved a satisfactory level of accuracy, especially for classes 0, 5, 6, and 7, where most of the predictions correspond to proper labels. At the same time, there is a noticeable number of classification errors between some neighboring classes, particularly classes 3 and 4, which may be due to the similarity of lexical expression between these categories. Despite its simplicity, logistic regression has demonstrated the ability to distinguish between basic psychoemotional states, but its effectiveness may be limited in cases of complex interclass overlaps [32].

Table 22 – Results of experiments with text encoding methods

Type of encoding method	Logistic Regression	LGBM	LSTM
Bag of Words (BoW)	75,77%	79,91%	-
TF-IDF	78,81%	79,24%	-
Word2Vec	70,49%	65,72%	78,47%
fastText	70,18%	66,36%	79,36%

In general, all these studies have shown that different text representation methods significantly impact the accuracy of machine learning models in psychoemotional state classification tasks [41]. Classical algorithms and ensembles, such as logistic regression and LGBM, show quite good results when using vectorization methods such

as TF-IDF and Bag of Words. However, their effectiveness is limited by the lack of context, which makes them less sensitive to semantic relations in the text.

TF-IDF/Bag-of-Words (BoW) are incompatible with LSTM because they destroy word sequence, and BiLSTM needs sequential input to work correctly. On the contrary, LSTM-based [50] models demonstrate higher accuracy, especially with methods such as Word2Vec and fastText, as they can take into account contextual dependencies and semantic nuances [42, 44]. This confirms that modern approaches to natural language processing are worth using for tasks that require a deeper understanding of the text.

The text contains a large number of pronouns and prepositions, the latter of which do not have a significant impact on the semantic load of the text. Therefore, in the future, it will be necessary to process all messages to use this dataset for training.

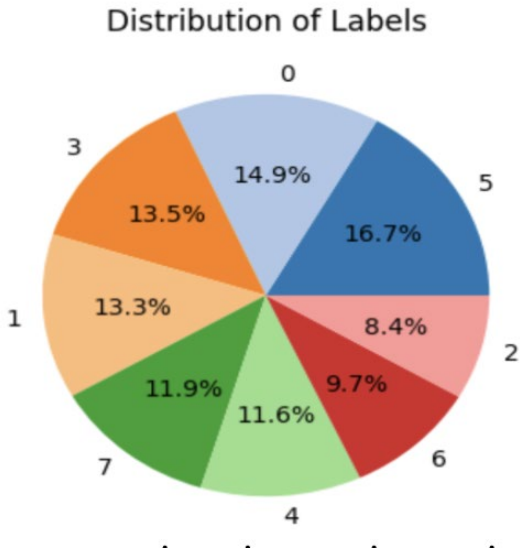


Figure 34 - Diagram showing the distribution of records by psycho-emotional states from the "reddit-mental-illness-82" dataset.

Before feeding texts into machine learning models, data from the dataset undergoes a thorough multi-stage pre-processing to reduce noise, increase the generalization ability of the models, and ensure a more correct representation of textual information. This processing includes the following steps:

1. Tokenization - splitting the text into separate components - tokens. These can be words, punctuation marks, or phrases. Depending on the chosen library (for example, nltk, spaCy, Stanza), tokenization can account for language features, which

is especially important for the Ukrainian language.

2. Eliminating stop words - removing general functional words (pronouns, prepositions, conjunctions, etc.) that do not carry a semantic load. For example: "and", "but", "it", "on", "in", etc. Stop words are determined by a predefined dictionary or contextually.

3. Lemmatization - reducing words to their basic (dictionary) form (lemma). For example: "said", "speak", "speak" → "speak". This reduces data variability and improves generalization.

4. Stemming (optional step) - truncation of words to their root. For example: "work", "labor", "working" → "worker". Although this approach is less accurate than lemmatization, it can be helpful in some classification cases.

5. Case normalization - converts all text characters to lowercase, which avoids duplication of the exact words with different spellings (e.g., "Depression" and "depression").

6. Removing punctuation, numbers, and non-alphabetic characters cleans up text from uninformative characters that may complicate or distort the analysis. This includes periods, commas, quotation marks, numbers, special characters, etc.

7. Formatting cleanup - removes unnecessary spaces, line breaks, tabs, etc. to unify the text structure.

8. Converting text to numeric representations - texts are converted into sequences of numbers according to a built dictionary or fixed-length vectors for further representation in machine learning models.

Converting text into a sequence of numbers according to the dictionary. This dataset can be used as input for a future machine learning model. To improve the performance of the models, we will experiment with tuning their hyperparameters. For example, XGBoost will test the optimal number of trees, tree depth, and node partitioning criteria.

To improve the performance of the models, experiments will be conducted with tuning their hyperparameters. For example, the optimal number of trees, tree depth, and node partitioning criteria will be tested in XGBoost.

For LSTM [54] and BERT [30], the optimal length of hidden states, the number of layers, the size of the text processing window, and the learning speed will be investigated. Special attention will be paid to regularization and anti-overfitting techniques such as Dropout and Batch Normalization.

Different vectorization methods will be tested to evaluate the impact of text representation on classification quality. For the models, the effectiveness of Bag of Words, TF-IDF, Word2Vec [28], and fastText will be investigated.

5.3. Utilization of ML libraries and platforms

In implementing the study, we used the Python programming language, which is currently one of the most widely used and effective in machine learning and natural language processing. This choice was made due to its open software environment, large community of users, and the availability of powerful tools for the full cycle of data processing, from text pre-processing to building and testing complex models. Several libraries were used to implement machine learning and deep learning algorithms, including scikit-learn, LightGBM, and TensorFlow. The scikit-learn library provided access to basic classification models, such as logistic regression, as well as tools for data partitioning, hyperparameter selection, and model quality assessment. LightGBM was used as a high-performance framework for gradient boosting on tree solutions, which allowed us to work efficiently with large amounts of data.

To build models based on recurrent neural networks, particularly LSTM, the TensorFlow library was used in combination with the high-level Keras API, which provided flexibility and scalability during modeling. The stage of text data pre-processing was implemented using the NLTK, spaCy libraries, and the re (regular expressions) module. They were used to normalize the text, including lowercase, cleaning from unwanted characters, removing stop words, and lemmatization. This approach ensured standardization of the input data, which is a critical step when working with text classifiers. To ensure a uniform representation of classes in the sample, we used the RandomUnderSampler method from the imblearn library, which

eliminated the imbalance of labels that negatively affects the quality of model training.

To implement the experiments, we chose the Kaggle cloud platform, which provided free access to hardware resources, including GPU acceleration (30 hours per week), as well as a pre-configured environment with all the necessary libraries [58]. This ensured efficient and fast computational performance, particularly when training computationally intensive models such as LSTM and BERT. In addition, integration with the data storage system and the platform's [74] user-friendly interface made it possible to organize a convenient environment for replicating results and analyzing model performance.

5.4. Description of experiments conducted

As mentioned in the previous sections, I decided to choose one model from each class of machine learning methods for comparison: logistic regression - classical, LightGBM - ensemble, bidirectional LSTM - neural networks, BERT and its variations: RoBerta, DeBerta - transformers. The choice of these algorithms for further research is based not only on the analysis of articles and relevant literature, but also on the selection of experiments in which these methods showed the best results.

Table 23 shows the results of the initial comparison of different machine learning methods grouped by algorithm type: classical, ensemble, and neural networks. For each method, the achieved classification accuracy in solving the task of determining a person's psycho-emotional state from text is indicated. Among the classical models, logistic regression demonstrated the highest accuracy (78.81%), which led to its selection for further experiments in this group. Other models, such as SVM with a linear kernel (77.33%) and a naive Bayesian classifier (73.04%), were inferior in quality, and K-NN was less suitable for this type of data (57.80%).

In the category of ensemble methods, the best result was shown by the LightGBM model (79.24%), which exceeded the performance of Random Forest (77.51%) and XGBoost (78.05%), demonstrating the effectiveness of tree-based gradient boosting in the context of text feature processing. Among the deep learning models, the

Table 23 - Comperison of methods, methodology an accuracy results

Class	Method	Accuracy
Classic	Logistic regression	78,81%
	SVM (kernel = linear)	77,33%
	K-NN (best k =11, metric = euclidian)	57,80%
	Naive-bayes	73,04%
Ensemble	Random Forest	77,51%
	XGBoost	78,05%
	LGBM	79,24%
Neural Network	Bidirectional LSTM with attention	75,13%
	CNN	74,43%

bidirectional LSTM with attention showed superior accuracy (75.13%) compared to the CNN [17] network (74.43%). This indicates the better ability of the LSTM model to capture sequential dependencies in the text, which is critical for emotional state analysis tasks.

Therefore, further research will focus on the most promising models of each group, which allows for a deeper analysis of their performance, stability, and generalizability. In order to improve the performance of the models, experiments [39] will be conducted to tune their hyperparameters. Particular attention will be paid to regularization and the use of anti-overfitting techniques such as Dropout and Batch Normalization.

Different vectorization methods will be tested to evaluate the impact of text representation on classification quality. For the models, the effectiveness of Bag of Words, TF-IDF, Word2Vec, and fastText will be investigated.

A separate set of experiments will be aimed at studying the role of text

preprocessing and the role of headlines in texts. We will test how the accuracy of the models changes when using only the main text or its combination with headlines. In the course of the experimental part of this study, a hyperparameter selection procedure [69] was applied to each of the selected machine and deep learning algorithms to optimize their performance. Figure 35 shows the list of hyperparameters for LGBM:

```
grid_params = {
    'max_depth': [3, 6, 9],
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1]
}
```

Figure 35 – Hyperparamether for LGBM

In the LightGBM model, the key hyperparameters that determine the learning efficiency and generalization capability of the model are max_depth, n_estimators, and learning_rate. The max_depth parameter sets the maximum depth of the decision tree and controls the complexity of each tree: lower values help reduce the risk of overfitting, while higher values allow the model to capture more complex patterns. n_estimators is responsible for the number of trees used in the model and directly affects its ability to learn - increasing this value improves accuracy but can lead to overfitting.

The learning_rate parameter determines the learning rate, i.e. the degree to which each individual tree contributes to the overall forecast; reducing it usually improves the model's ability to generalize, but requires more trees. Joint optimization of these parameters is an important part of the model tuning process to achieve a balance between accuracy and overfitting. The following results were obtained from this experiment presented in Table 24.

Table 24 shows that the highest accuracy (79.91%) was demonstrated by the model using the BoW method with learning_rate = 0.1, max_depth = 6, and n_estimators = 300. The TF-IDF method showed similar performance (79.24%) with a deeper tree (max_depth = 9), but slightly lower overall performance. Despite similar

Table 24 - Results of hyperparameter selection and text representation method for LGBM

Encoding methods	Hyperparamethes			Accuracy
	learning rate	max depth	n estimators	
Bag of Words (BoW)	0,1	6	300	79,91%
TF-IDF	0,1	9	200	79,24%
Word2Vec	0,1	6	300	65,72%
fastText	0,1	6	300	66,36%

hyperparameters, the Word2Vec and fastText methods showed significantly lower accuracy (65.72% and 66.36%, respectively).

These results indicate that classical frequency-based text representation methods (BoW, TF-IDF) are more efficient for working with a tree-based gradient boosting model such as LightGBM. This is because such methods generate highly sparse but interpretable feature vectors that are well matched to the tree structure. Instead, contextual vector representations (Word2Vec, fastText) perform better in neural architectures that are able to capture complex semantic relations, but in the case of tree-based models, they do not provide advantages and may even reduce accuracy due to the loss of specific frequency patterns.

Classification Report (Best Model):

	precision	recall	f1-score	support
0	0.86	0.82	0.84	790
1	0.79	0.77	0.78	807
2	0.82	0.71	0.76	858
3	0.74	0.71	0.73	826
4	0.61	0.79	0.69	822
5	0.88	0.97	0.92	751
6	0.89	0.83	0.86	822
7	0.87	0.81	0.84	862
accuracy			0.80	6538
macro avg	0.81	0.80	0.80	6538
weighted avg	0.81	0.80	0.80	6538

Figure 36 - Classification report for LGBM

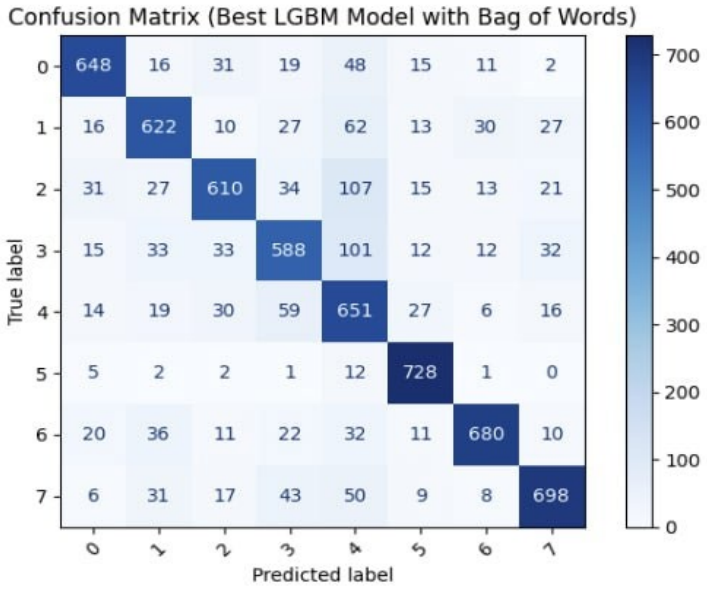


Figure 37 - Confusion matrix for LGBM

The results of classifying psycho-emotional states using the LightGBM model in combination with Bag of Words [82] vectorization show an overall accuracy of 80%, which indicates its effectiveness in multi-class text recognition. The highest classification quality was recorded for classes 5, 6, and 7 (f1-score 0.92, 0.86, and 0.84, respectively), while the lowest performance was observed for class 4 (f1 = 0.69), indicating the difficulty of recognizing it within this model. The average metrics (macro and weighted average f1-score) of 0.80 confirm the stable overall performance, but the presence of cross-validation errors in the mismatch matrix indicates the potential for improving accuracy by using more context-sensitive models.

The search for optimal values of hyperparameters for LSTMs is based on a heuristic strategy used in Optuna - Tree-structured Parzen Estimator (TPE). This is a Bayesian optimization approach that is based on building probability density models to efficiently select the next candidates. TPE allows you to focus on those areas of the hyperparameter space where the loss function performs best, making the search faster and more efficient than the classic Grid Search or Random Search.

When setting up a model using this library, the objective function defines the search space for hyperparameters for a recurrent neural network using an LSTM layer.

```
# Optuna Objective Function
def objective(trial):
    lstm_units = trial.suggest_int('lstm_units', 64, 256, step=32)
    dropout_rate = trial.suggest_float('dropout_rate', 0.1, 0.5, step=0.1)
    learning_rate = trial.suggest_loguniform('learning_rate', 1e-4, 1e-2)
```

Figure 38 - Hyperparamethers for LSTM

•lstm_units is responsible for the number of neurons in the LSTM layer — it controls the model's memory capacity and its ability to model dependencies in sequences; increasing the number of units can improve the model's ability to represent complex patterns.

•dropout_rate is the probability of randomly turning off neurons during training to prevent overfitting; selecting the optimal value allows you to find a balance between generalization and loss of useful information.

•learning_rate sets the rate at which the network weights are updated during optimization.

Table 25 - Results of hyperparameter selection and text representation method for Bidirectional LSTM with attention

Encoding methods	Hyperparamethes			Accuracy
	lstm units	dropout rate	learning rate	
Word2Vec	64	0,1	0,006717	78,48%
fastText	128	0,4	0,00812	79,36%

Table 25 shows the results of optimizing the Bidirectional LSTM model with an attention mechanism for two vectorization methods: Word2Vec and fastText. The highest accuracy (79.36%) is shown when using fastText, which is due to its ability to take into account the morphological features of words. Other vectorization methods (TF-IDF, BoW) were not used due to their limited ability to convey semantics and context, which is critical for this model.

The summary of table presents the results of this experiment, also presenting the results of logistic regression with different embeddings. The results of the experiments

showed variability in classification accuracy depending on the chosen method of text vectorization when using logistic regression. The TF-IDF method demonstrated the highest efficiency, achieving an accuracy of 78.81%, confirming its ability to effectively take into account the importance of individual terms in the context of the corpus [107]. The bag-of-words (BoW) method also showed a noticeable level of performance (75.77%), while Word2Vec and fastText showed lower results — 70.49% and 70.18%, respectively. This may indicate the limitations of the linear model in fully utilizing the semantic information embedded in dense vector representations.

Classification Report:

	precision	recall	f1-score	support
0	0.80	0.83	0.82	790
1	0.77	0.74	0.75	807
2	0.79	0.70	0.74	858
3	0.73	0.70	0.72	826
4	0.63	0.77	0.69	822
5	0.89	0.98	0.93	751
6	0.90	0.81	0.85	822
7	0.84	0.79	0.81	862
accuracy			0.79	6538
macro avg	0.79	0.79	0.79	6538
weighted avg	0.79	0.79	0.79	6538

Figure 39 - Classification report for logistic regression

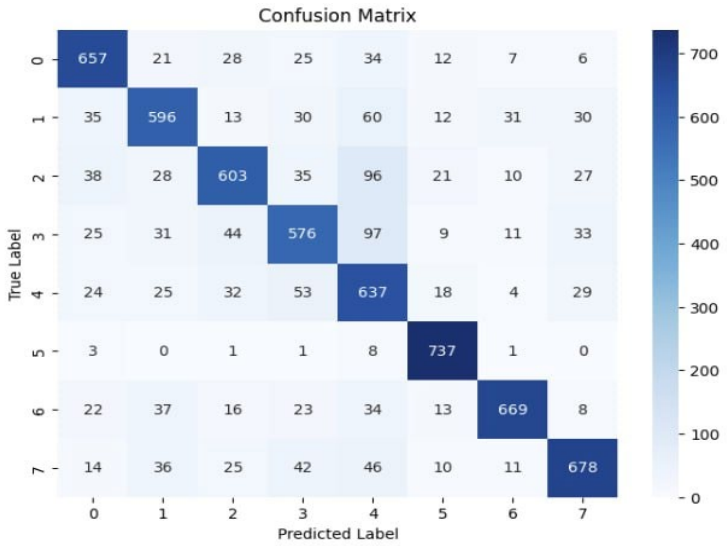


Figure 40 - Confusion matrix for logistic regression

For BERT, there is no point in using Bag of Words, TF-IDF, Word2Vec, or

fastText, as it already works with its own contextual embeddings. And its accuracy was 81.35% when trained on 7 epochs.

```
Epoch 1/7: 100%|██████████| 1635/1635 [19:34<00:00,  1.39it/s]
Epoch 1/7 - Loss: 0.7709, Accuracy: 0.7505
Epoch 2/7: 100%|██████████| 1635/1635 [19:35<00:00,  1.39it/s]
Epoch 2/7 - Loss: 0.4871, Accuracy: 0.8454
Epoch 3/7: 100%|██████████| 1635/1635 [19:35<00:00,  1.39it/s]
Epoch 3/7 - Loss: 0.3413, Accuracy: 0.8906
Epoch 4/7: 100%|██████████| 1635/1635 [19:34<00:00,  1.39it/s]
Epoch 4/7 - Loss: 0.2073, Accuracy: 0.9370
Epoch 5/7: 100%|██████████| 1635/1635 [19:34<00:00,  1.39it/s]
Epoch 5/7 - Loss: 0.1197, Accuracy: 0.9667
Epoch 6/7: 100%|██████████| 1635/1635 [19:32<00:00,  1.39it/s]
Epoch 6/7 - Loss: 0.0703, Accuracy: 0.9820
Epoch 7/7: 100%|██████████| 1635/1635 [19:32<00:00,  1.39it/s]
Epoch 7/7 - Loss: 0.0442, Accuracy: 0.9896
Evaluating: 100%|██████████| 409/409 [01:38<00:00,  4.14it/s]BERT Transformer Model Evaluation:
Accuracy: 0.8135515448149281
```

Figure 41 – Training process BERT

	precision	recall	f1-score	support
0	0.88	0.84	0.85	790
1	0.75	0.79	0.77	807
2	0.79	0.76	0.78	858
3	0.76	0.72	0.74	826
4	0.67	0.74	0.71	822
5	0.99	0.97	0.98	751
6	0.87	0.86	0.87	822
7	0.83	0.84	0.84	862
accuracy			0.81	6538
macro avg	0.82	0.82	0.82	6538
weighted avg	0.82	0.81	0.81	6538

Figure 42 - Classification report of BERT

The BERT model, which was pre-trained on a large natural language corpus and retrained on a specific classification task, demonstrated higher quality indicators across all key metrics. In particular, its overall accuracy was 81.35%, which is 2.54% higher than the logistic regression model (79%), 1.44% higher than LGBM, and 1.99% higher than LSTM. Similarly, the macro-average value of the F1-measure is 0.82 vs. 0.79, and the weighted average is 0.81. Improvements are observed in each of the eight classes corresponding to individual psychoemotional states. The largest increase in the F1-measure was observed for class 5, where BERT reached a value of 0.98, indicating that the transformer-type model is better able to identify emotionally expressed messages. In the most difficult to classify class 4, the BERT model also outperformed other models, with an F1-measure of 0.71. Summarizing, we can say that the BERT model

is a much more effective approach for the task of classifying psycho-emotional state based on text, especially in the context of complex linguistic structures and ambiguity of statements.

While analyzing the text, I noticed that in each of them, you can distinguish the title before the colon. In fact, I had an idea that it might influence the classification of the text.

	title	\
0	fantasize fp	
1	support subreddit people mental illness	
2	scared psychotic symptom	
4	feel like sick animal need put	
5	constantly check ocd subreddit compulsion	
...	...	
52637	get diagnose pure finding difficult come term ...	
52638	read propaganda	
52639	need help keep thought straight maintain figur...	
52640	anyone else feel like none experience valid	
52641	wish luck work today	

	main_text
0	fantasize fp attract fp romantic way fantasize...
1	post infuriate uneven floor tile satisfy candy...
2	I m try keep remind I m think believe be not r...
4	cat year end life lose mental faculty lethargi...
5	kid thought I m constantly check ocd subreddit...
...	...
52637	spend life worry weird different hate judge el...
52638	want learn tactic psychology go along history ...
52639	I ve always crap math abstract concept can not...
52640	I ve lot people tell anxiety do not think they...
52641	oh god anxiety kill today I m work get ready c...

Figure 43 - An example of division into main text and title

Then only the main text was sent to the input of four models to test the hypothesis.

And after conducting this experiment, it was found that the accuracy of the models is significantly reduced when using only the main text without headings. Logistic regression - 74.16%, LGBM - 72.72%, BERT - 78.22%, LSTM - 69.36%

Thus, the experiment with extracting headlines from text messages showed a significant decrease in classification accuracy for all the models studied, including logistic regression, LGBM, BERT [81], and LSTM [53]. This indicates that headlines play a significant role in the process of modeling psychoemotional state, probably due to their high information content and generalized nature, which facilitates the identification of the main content of the message. Thus, headlines should be considered as a separate significant component of the input text when building automatic classification models.

In the preliminary experiments, the BERT model showed the best classification accuracy (81.35%), but when analyzing the articles, I saw many variations of this model that could show better results, given their parameters. Therefore, I chose 2 models for further experiments: RoBerta and DeBerta.

For both models, the input sequence size was limited to 256 tokens (max_len=256), which is a standard practice to ensure a balance between computational efficiency and completeness of the input information. AdamW was chosen as the optimizer with an initial learning rate of 2e-5, which is consistent with the recommendations for fine-tuning transformer models [8485]. To stabilize the learning process, the get_linear_schedule_with_warmup learning rate scheduler without warmup steps was used, and this ensured a linear decrease in the learning rate during all epochs (their number was 5). The batch size (batch_size = 16) was chosen taking into account the hardware resources and the model size, which allows efficient data processing without memory overload.

Looking at Figures 54 and 55, we can see a significant improvement in accuracy. Now, Roberta's classification accuracy is 82.75% and Deberta's is 83.07%, and there is an improvement across all classes and metrics compared to the original Bert model

```
Epoch 1/5: 100%|██████████| 1635/1635 [27:42<00:00,  1.02s/it]
Epoch 1/5 - Loss: 0.9175, Training Accuracy: 0.6828
Epoch 2/5: 100%|██████████| 1635/1635 [27:58<00:00,  1.03s/it]
Epoch 2/5 - Loss: 0.5628, Training Accuracy: 0.8210
Epoch 3/5: 100%|██████████| 1635/1635 [27:58<00:00,  1.03s/it]
Epoch 3/5 - Loss: 0.4525, Training Accuracy: 0.8579
Epoch 4/5: 100%|██████████| 1635/1635 [27:59<00:00,  1.03s/it]
Epoch 4/5 - Loss: 0.3598, Training Accuracy: 0.8868
Epoch 5/5: 100%|██████████| 1635/1635 [27:58<00:00,  1.03s/it]
Epoch 5/5 - Loss: 0.2855, Training Accuracy: 0.9131
Evaluating: 100%|██████████| 409/409 [02:24<00:00,  2.83it/s]DeBERTa-v3 Model Evaluation:
Test Accuracy: 0.8307
              precision    recall  f1-score   support

           0       0.90      0.86      0.88       790
           1       0.77      0.80      0.78       807
           2       0.81      0.76      0.79       858
           3       0.79      0.74      0.77       826
           4       0.69      0.79      0.73       822
           5       0.99      0.98      0.99       751
           6       0.87      0.88      0.88       822
           7       0.86      0.85      0.86       862

    accuracy                           0.83      6538
   macro avg       0.84      0.83      0.83      6538
weighted avg       0.83      0.83      0.83      6538
```

Figure 44 - Training process and classification report for DeBerta

```
Some weights of RobertaForSequenceClassification were not initialized from the model checkpoint at roberta-base
e.weight', 'classifier.out_proj.bias', 'classifier.out_proj.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Epoch 1/5: 100%|██████████| 1635/1635 [18:48<00:00,  1.45it/s]
Epoch 1/5 - Loss: 0.7577, Training Accuracy: 0.7511
Epoch 2/5: 100%|██████████| 1635/1635 [18:49<00:00,  1.45it/s]
Epoch 2/5 - Loss: 0.5151, Training Accuracy: 0.8356
Epoch 3/5: 100%|██████████| 1635/1635 [18:49<00:00,  1.45it/s]
Epoch 3/5 - Loss: 0.4060, Training Accuracy: 0.8696
Epoch 4/5: 100%|██████████| 1635/1635 [18:49<00:00,  1.45it/s]
Epoch 4/5 - Loss: 0.3078, Training Accuracy: 0.9023
Epoch 5/5: 100%|██████████| 1635/1635 [18:49<00:00,  1.45it/s]
Epoch 5/5 - Loss: 0.2350, Training Accuracy: 0.9283
Evaluating: 100%|██████████| 409/409 [01:22<00:00,  4.93it/s]
RoBERTa Model Evaluation:
Test Accuracy: 0.8275
              precision    recall  f1-score   support

           0       0.88      0.85      0.87       790
           1       0.78      0.78      0.78       807
           2       0.79      0.78      0.79       858
           3       0.77      0.75      0.76       826
           4       0.70      0.76      0.73       822
           5       0.99      0.98      0.98       751
           6       0.88      0.87      0.87       822
           7       0.85      0.86      0.85       862

    accuracy                           0.83      6538
   macro avg       0.83      0.83      0.83      6538
weighted avg       0.83      0.83      0.83      6538
```

Figure 45 - Training process and classification report for RoBerta

5.5 Development of an ensemble-based model on transformers

Within the scope of the study, an ensemble approach combining two advanced transformer architectures — DeBERTa-v3-base and RoBERTa-base — was implemented to improve the quality of psycho-emotional state classification. This strategy involves combining the predictions of individual models to obtain a more stable and consistent result.

The principle of ensemble construction is the soft voting method, which consists of averaging the logits (normalized pre-activation values) of both models. After that, the class with the highest average logit value is selected for each example. This approach allows for the degree of confidence of each model for each class to be taken into account, which makes the decision more balanced compared to hard voting, which only takes into account the final predictions.

Ensemble learning was chosen because of the following advantages:

•Compensation for model weaknesses: RoBERTa works well with large amounts of text thanks to pre-training on a large corpus without segments, while DeBERTa processes context more efficiently through improved positional encoding.

•Classification reliability: combining two powerful models reduces the risk of overfitting characteristic of a single model and improves the generalization ability of the system.

Thus, the implemented ensemble of DeBERTa and RoBERTa achieved higher classification accuracy, namely 83.97%, surpassing all models, and ensured the model's stability to the variability of text data in the context of psycho-emotional analysis, as can be seen in Figures 56 and 57.

	precision	recall	f1-score	support
0	0.90	0.87	0.88	790
1	0.79	0.80	0.80	807
2	0.83	0.79	0.81	858
3	0.79	0.77	0.78	826
4	0.71	0.80	0.75	822
5	0.99	0.98	0.98	751
6	0.89	0.88	0.88	822
7	0.85	0.85	0.85	862
accuracy			0.84	6538
macro avg	0.84	0.84	0.84	6538
weighted avg	0.84	0.84	0.84	6538

Figure 46 - Classification report для ensemble models

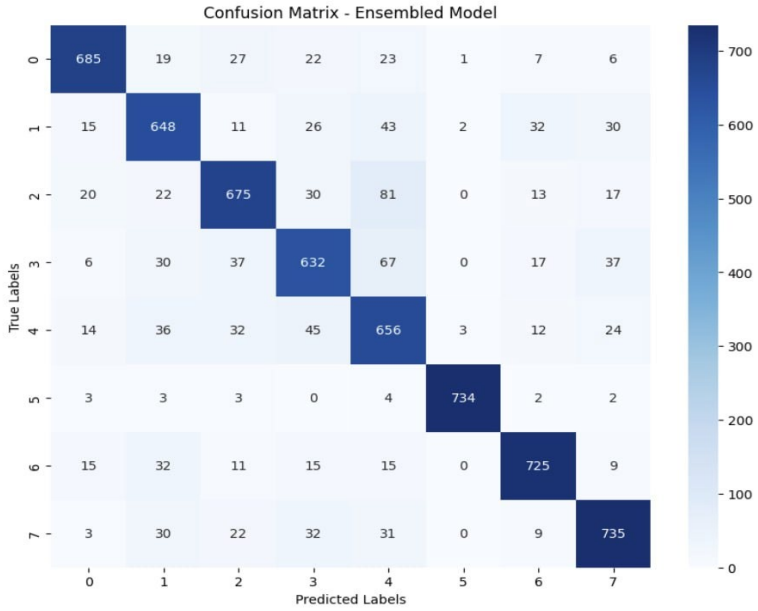


Figure 47 - Confusion matrix for ensemble models

5.6. The importance of stop words in model predictions

In the context of analyzing psycho-emotional states, experiments involving the removal or retention of stop words are necessary to determine their impact on the accuracy of mental context recognition [46]. Although stop words are often considered uninformative in thematic classification tasks, in cases of psycholinguistic analysis, they can carry significant semantic or stylistic information [101].

For this experiment, the best model from the previous classes was selected—ensemble RoBerta–DeBerta. Figure 58 show the training results.

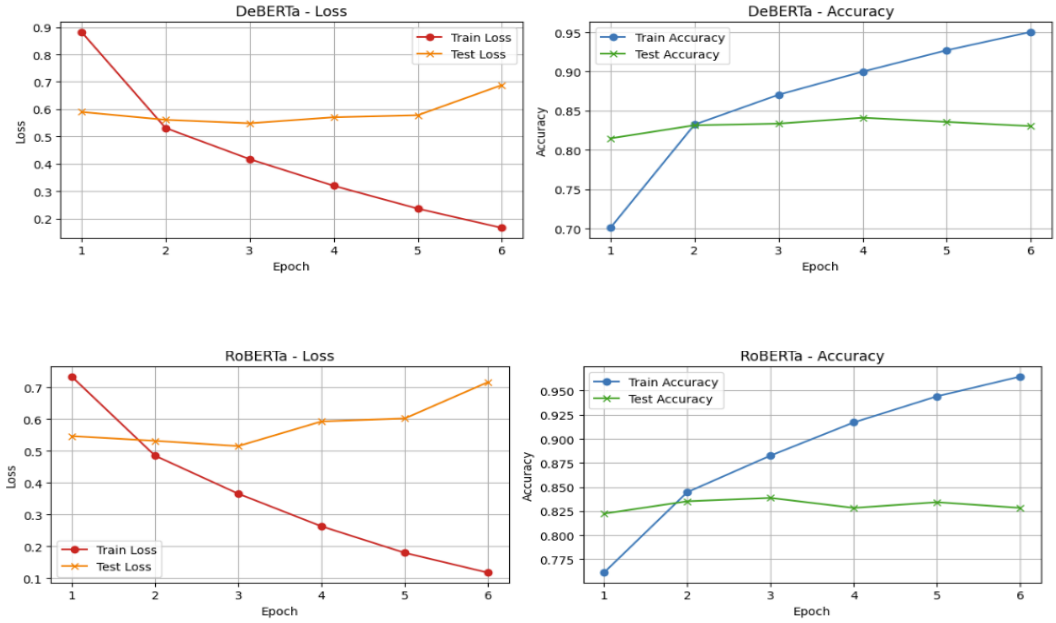


Figure 48 – Learning curves for RoBerta and DeBerta models

During the training of the RoBERTa and DeBERTa models, behavior characteristic of large transformer architectures is observed, including a sharp decrease in training error and partial signs of overfitting [46]. The graphs show that training loss decreases rapidly in both models, reaching very low values in the sixth epoch. Thus, both models are capable of reproducing the training data very accurately [81], i.e., they "memorize" it well. However, the test loss, which decreases in the early stages of training, begins to increase after 3–4 epochs, indicating a decrease in the model's ability to generalize — a typical symptom of overfitting.

However, it is important to note that despite the increase in test loss, the accuracy on the test set remains stable (approximately 84% for both models). This shows that

```
DeBERTa Evaluation:
Accuracy: 0.8377
              precision    recall  f1-score   support

           0       0.91      0.86      0.88       790
           1       0.75      0.83      0.79       807
           2       0.82      0.77      0.79       858
           3       0.79      0.76      0.77       826
           4       0.72      0.80      0.76       822
           5       0.99      0.98      0.99       751
           6       0.89      0.88      0.89       822
           7       0.87      0.84      0.85       862

    accuracy                           0.84      6538
   macro avg       0.84      0.84      0.84      6538
weighted avg       0.84      0.84      0.84      6538


RoBERTa Evaluation:
Accuracy: 0.8383
              precision    recall  f1-score   support

           0       0.91      0.87      0.89       790
           1       0.77      0.81      0.79       807
           2       0.82      0.76      0.79       858
           3       0.80      0.76      0.78       826
           4       0.72      0.78      0.75       822
           5       0.99      0.99      0.99       751
           6       0.88      0.89      0.89       822
           7       0.85      0.86      0.85       862

    accuracy                           0.84      6538
   macro avg       0.84      0.84      0.84      6538
weighted avg       0.84      0.84      0.84      6538

Ensemble Evaluating: 100%|██████████| 409/409 [03:28<00:00,
Ensembled Model Evaluation:
Test Accuracy: 0.8507
              precision    recall  f1-score   support

           0       0.92      0.88      0.90       790
           1       0.78      0.84      0.81       807
           2       0.84      0.79      0.81       858
           3       0.81      0.78      0.79       826
           4       0.73      0.80      0.77       822
           5       0.99      0.99      0.99       751
           6       0.91      0.89      0.90       822
           7       0.86      0.86      0.86       862

    accuracy                           0.85      6538
   macro avg       0.85      0.85      0.85      6538
weighted avg       0.85      0.85      0.85      6538
```

Figure 49 - Classification report for RoBerta, DeBerta and ensemble methods

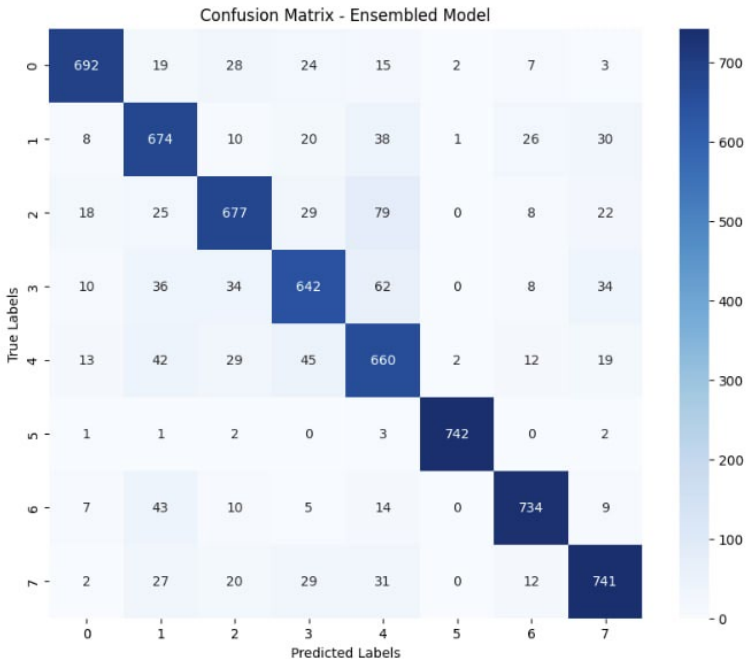


Figure 50 - Confusion matrix for ensemble models, without stop words

the models still demonstrate good generalization ability, even despite overfitting. To

prevent a deterioration in generalization quality, it was decided to keep the model weights at epoch 5, when the test accuracy was stable and the increase in test loss was not yet critical. This approach can be considered an element of early stopping, which is an effective strategy for combating overfitting.

Improvements are observed both in individual models (DeBerta by 0.7% from the previous experiment, RoBerta by 1.08%) and in ensembling by 1.1%. This indicates the importance of stop words in further classification.

Conclusions

This section provides a comprehensive analysis of the impact of different approaches to text representation on the effectiveness of models for classifying psycho-emotional states. Experiments have shown that the accuracy of results depends largely on the choice of vectorization and preprocessing methods, as well as on the model architecture. Classic machine learning methods demonstrated limited ability to model semantic context, while modern neural architectures, especially transformers, provided a significant improvement in classification quality thanks to deeper contextual text processing. The use of recurrent networks with pre-trained embeddings (Word2Vec, fastText) also increased the sensitivity of models to semantic nuances, but was inferior to the results of transformer approaches. Additionally, it was found that the presence of titles in text messages plays an important role in improving classification results, as they contain a concentrated and informative semantic component that contributes to more accurate recognition of the emotional context. A comparative analysis of transformer models confirmed the superiority of the RoBERTa and DeBERTa architectures over the basic BERT model. Further accuracy improvements were achieved by applying an ensemble method using soft voting and without removing stop words, which allowed us to effectively combine the strengths of both models.

Thus, the results clearly demonstrate the feasibility of using context-sensitive transformer models, as well as their ensemble, to improve the quality of automated analysis of psycho-emotional state based on text data. In the future, it is worth exploring a multi-format approach that combines text data with voice [9-11] or visual

signals [11-16] for a deeper analysis of the psycho-emotional state. It is also important to optimize preprocessing, taking into account the peculiarities of language and text style. Another promising direction is the improvement of model ensemble methods.

The study showed that the choice of text representation method and machine learning algorithm has a significant impact on the accuracy of psycho-emotional [35] state classification. In particular, traditional methods such as logistic regression and XGBoost showed good results when using TF-IDF, but have limited ability to recognize contextual relationships, which leads to a decrease in their accuracy. In contrast, LSTM-based models proved to be more context-sensitive due to their architecture [83], confirming their ability to better process text data. Of particular importance is the fact that BERT, thanks to its contextual embeddings, does not require classical vectorization methods, making it a powerful tool for text analysis tasks. Its accuracy of 81.35% demonstrates the effectiveness of transformer models in classifying psycho-emotional states.

Summary and conclusions.

This monograph presents a comprehensive study of methods for the intellectual analysis of Ukrainian-language texts with the aim of identifying emotional markers and diagnosing signs of stress. The work covers both theoretical and applied aspects of natural language processing, taking into account the linguistic, psycholinguistic, and technical features of the Ukrainian language.

The theoretical section examines the basic concepts, classifications of emotions, the structure of emotional perception of text, and key approaches to the automatic determination of emotions and psycho-emotional tension. Particular attention is paid to the specifics of the Ukrainian language as an object of NLP research.

At the data preparation stage, approaches to preprocessing, annotation of emotional content, and the use of available linguistic resources were justified. Data corpora annotated according to emotional criteria were constructed, which became the basis for further machine analysis.

As part of the analysis of emotional markers, methods for identifying key lexemes, thematic modeling, and determining the tone of the text were proposed. Approaches to identifying signs of stress based on the synthesis of psycholinguistic patterns and statistical indicators were developed.

At the experimental level, a comparative analysis of machine learning models was conducted, in particular based on transformer architectures (BERT, RoBERTa), as well as hybrid methods that integrate syntagmatic and paradigmatic connections in the text. The classification results showed the high efficiency of models adapted to the specifics of the Ukrainian language, with an accuracy of over 83% in determining emotional state.

The practical result of the research was the creation of a prototype emotion analysis system, which was implemented using modern NLP libraries. The evaluation of the system's effectiveness confirmed the feasibility of using combined approaches to emotional analysis.

The conducted study has laid a solid methodological foundation for the automated analysis of the psycho-emotional content of Ukrainian-language texts. However, several important areas require further development and deepening. Among the key promising directions for future research, the following deserve particular attention.

First, expanding the corpus of emotionally labeled Ukrainian texts remains an urgent task. The accumulation of high-quality, annotated data from diverse domains (e.g., media materials, educational texts, private communication) will significantly enhance the generalization ability of models and ensure their robustness in cross-domain applications.

Second, it is essential to explore in greater depth the relationships between lexical, syntactic, and stylistic features of the text and the psycho-emotional state of the author. Such exploration will form the basis for modeling more complex cognitive states, such as anxiety, depressive tendencies, or professional burnout.

Third, there is significant potential in integrating textual data with other modalities [25], including audio [64], video [65], or behavioral signals. This integration opens the way for the development of multimodal emotion recognition systems [26],

which are especially relevant in the contexts of remote learning, digital psychodiagnostics, and smart communication.

The fourth direction involves further improvement of transformer-based models, particularly those built on BERT-like architectures, while considering the morphological and syntactic specifics of the Ukrainian language. In the long term, the development of custom transformer-based language models trained from scratch on Ukrainian corpora is a highly desirable objective [29].

Fifth, it is crucial to integrate the developed methods into applied services for real-time emotional state monitoring, particularly in the fields of education, mental health care, HR analytics, and information security.

Another critically important area is ensuring the ethical use of such systems, including the protection of user privacy, transparency and explainability of model decisions, and the prevention of misuse related to the automated interpretation of emotional signals.

Thus, continued research in this field holds great potential to significantly expand both the practical capabilities of Ukrainian-language text analysis and the theoretical foundations for building adaptive, context-sensitive intelligent natural language processing systems.

Verweise / References

1. A. Vaswani et al., "Attention is all you need", in Advances in neural information processing systems, 2017, pp. 5998-6008.
2. "3 Apps for Mental Health". App Development, Resources, Artificial Intelligence. URL: https://www.stonesouptech.com/3-apps-for-mental-health/ (available date: 25.09.2024).
3. "How much does it cost to develop an app like Wysa?" (Feb 1, 2024). URL: https://www.simublade.com/blogs/cost-to-develop-a-mental-health-app-like-wysa/ (available date: 25.09.2024).
4. "Психічне здоров'я – одне з параметрів, що визначає якість життя населення". Департамент охорони здоров'я тернопільської облдерж адміністрації. URL: https://uozter.gov.ua/ua/pages/252 (available date: 20.09.2024).
5. A General Approach to Preprocessing Text Data // Machine learning, Data Science, big Data, Analytics, AI. URl: https://www.kdnuggets.com/2017/12/generalapproachpreprocessing-text-data.html
6. Abdulmajeed N. Q. A review on voice pathology: taxonomy, diagnosis, medical procedures and detection techniques, open challenges, limitations, and recommendations for future directions / N. Q. Abdulmajeed, B. Al-Khateeb, M. A. Mohammed // Journal of Intelligent Systems. — 2022. — Vol. 31, No. 1. — P. 855–875.
7. Abdulmajeed, N. Q., Al-Khateeb, B., and Mohammed, M. A. (2022). A review on voice pathology: taxonomy, diagnosis, medical procedures and detection techniques, open challenges, limitations, and recommendations for future directions. J. Intell. Syst. 31, 855–875. doi: 10.1515/jisys-2022-0058
8. Abhirami V. A. "Softmax vs LogSoftmax". Medium, 2021. URL: https://medium.com/@AbhiramiVS/softmax-vs-logsoftmax-eb94254445a2 (дата звернення: 21.11.2024).

9. Al-Dhief F. T. A survey of voice pathology surveillance systems based on internet of things and machine learning algorithms / F. T. Al-Dhief, N. M. A. Latiff, N. N. N. Abd. Malik, [et al.] // IEEE Access. — 2020. — Vol. 8. — P. 64514–64533.
10. Al-Dhief F. T. Voice pathology detection and classification by adopting online sequential extreme learning machine / F. T. Al-Dhief, M. M. Baki, N. M. A. Latiff, [et al.] // IEEE Access. — 2021. — Vol. 9. — P. 77293–77306.
11. Al-Dhief, F. T., Baki, M. M., Latiff, N. M., Malik, N. N., Salim, N. S., Albader, M. A., et al. (2021). Voice pathology detection and classification by adopting online sequential extreme learning machine. IEEE Access 9, 77293–77306. doi: 10.1109/ACCESS.2021.3082565
12. Al-Dhief, F. T., Latiff, N. M., Malik, N. N., Salim, N. S., Baki, M. M., Albadr, M. A., et al. (2020). A survey of voice pathology surveillance systems based on internet of things and machine learning algorithms. IEEE Access 8, 64514–64533. doi: 10.1109/ACCESS.2020.2984925
13. Alhussein M. Automatic voice pathology monitoring using parallel deep models for smart healthcare / M. Alhussein, G. Muhammad // IEEE Access. — 2019. — Vol. 7. — P. 46474–46479.
14. Alhussein M. Voice pathology detection using deep learning on mobile healthcare framework / M. Alhussein, G. Muhammad // IEEE Access. — 2018. — Vol. 6. — P. 41034–41041.
15. Alhussein, M., and Muhammad, G. (2018). Voice pathology detection using deep learning on mobile healthcare framework. IEEE Access 6, 41034–41041. doi: 10.1109/ACCESS.2018.2856238
16. Alhussein, M., and Muhammad, G. (2019). Automatic voice pathology monitoring using parallel deep models for smart healthcare. IEEE Access 7, 46474–46479. doi: 10.1109/ACCESS.2019.2905597
17. Alzubaidi, Laith, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions." Journal of big Data 8

(2021): 1-74.

18. Balit, E., & Chadli, A. (2020). GMFNet: Gated multimodal fusion network for visible-thermal semantic segmentation. In Proceedings of the 16th European Conference on Computer Vision (pp. 1–4).
19. Balit, E., & Chadli, A. (2020). GMFNet: Gated multimodal fusion network for visible-thermal semantic segmentation. In Proceedings of the 16th European Conference on Computer Vision (pp. 1–4).
20. Baracho R.M., Bax M., Ferreira L.G., Silva G.C. Sentiment analysis in social networks. Sentiment analysis and opinion mining. 2012. № 1. P. 115–125.
21. Basystiuk, O., Melnykova, N. (2023, October). Multimodal Learning Analytics: An Overview of the Data Collection Methodology. In 2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT) (pp. 1-4). IEEE.
22. Basystiuk, O.; Melnykova, N.; Rybchak, Z. Multimodal Learning Analytics: An Overview of the Data Collection Methodology. In Proceedings of the 2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT), Lviv, Ukraine, 19–21 October 2023.
23. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O., 2019. nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 .
24. Cao, F., Shi, T., Han, K., Wang, P., & An, W. (2023). RDFM: Robust deep feature matching for multi-modal remote sensing images. IEEE Geoscience and Remote Sensing Letters.
25. Cheng S-T, Lyu Y-J, Teng C. Image-Based Nutritional Advisory System: Employing Multimodal Deep Learning for Food Classification and Nutritional Analysis. Applied Sciences. 2025; 15(9):4911. https://doi.org/10.3390/app15094911
26. CM Transactions on Asian and Low-Resource Language Information Processing, vol. 23, 2024, pp. 1 - 20
27. Daxin Tan, Liqun Deng, Yu Ting Yeung, Xin Jiang, Xiao Chen,and Tan Lee,

"Editspeech: A text based speech editing systemusing partial inference and bidirectional fusion," arXiv preprintarXiv:2107.01554, 2021.

28. Derry Jatnikaa, Moch Arif Bijaksanaa, Arie Ardiyanti Suryania. "Word2Vec Model Analysis for Semantic Similarities in English Words". 4th International Conference on Computer Science and Computational Intelligence 2019, (ICCSCI), 12-13 September 2019
29. Deruty, E. (2025) Intuitive understanding of MFCCs. The mel frequency cepstral coefficients. Medium.
30. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.48550/arXiv.1810.04805
31. Duda, S. (2025) Urban environmental audio classification using mel spectrograms. Medium. Available at: https://scottmduda.medium.com/urban-environmental-audio-classification-using-mel-spectrograms-706ee6f8dcc1
32. Greeshma Gogula. "Mental Illness Detection Using NLP". California State University, Northridge: 2024-08-28, pp.1-86.
33. Havryliuk, M., Dumyn, I., Vovk, O. (2023). Extraction of Structural Elements of the Text Using Pragmatic Features for the Nomenclature of Cases Verification. In: Hu, Z., Wang, Y., He, M. (eds) Advances in Intelligent Systems, Computer Science and Digital Economics IV. CSDEIS 2022. Lecture Notes on Data Engineering and Communications Technologies, vol 158. Springer, Cham. https://doi.org/10.1007/978-3-031-24475-9_57
34. Hossain M. S. Healthcare big data voice pathology assessment framework / M. S. Hossain, G. Muhammad // IEEE Access. — 2016. — Vol. 4. — P. 7806–7815.
35. Hossain, M. S., and Muhammad, G. (2016). Healthcare big data voice pathology assessment framework. IEEE Access 4, 7806–7815. doi: 10.1109/ACCESS.2016.2626316

36. Ian Goodfellow, Yoshua Bengio, Aaron Courville, and YoshuaBengio, Deep learning, vol. 1, MIT press Cambridge, 2016
37. Index.dev. (2024, April 16). Comparing unimodal vs. multimodal models in generative AI. Index.dev Blog. https://www.index.dev/blog/comparing-unimodal-vs-multimodal-models
38. Iqra Ameer, Muhammad Arif, Grigori Sidorov, Helena Gómez-Adorno, Alexander F. Gelbukh. "Mental Illness Classification on Social Media Texts using Deep Learning and Transfer Learning". 2022, pp.1 – 12. CoRR abs/2207.01012.
39. Islam, R., and Tarique, M. (2022). A novel convolutional neural network based dysphonic voice detection algorithm using chromagram. Int. J. Electr. Comput. Eng. 12:5511. doi: 10.11591/ijece.v12i5.pp5511-5518
40. Jaafar, N.; Lachiri, Z. Multimodal fusion methods with deep neural networks and meta-information for aggression detection in surveillance. Expert Syst. Appl. 2022, 211, 118523.
41. Jean Nyandwi. "The Transformer Blueprint: A Holistic Guide to the Transformer Neural Network Architecture". AI Research Blog, 2023. URL: https://deeprevision.github.io/posts/001-transformer/ (дата звернення: 21.11.2024).
42. Jiang, Q., Chi, Z., Ma, X., Mao, Q., Yang, Y., & Tang, J. (2025). Rebalanced Multimodal Learning with Data-aware Unimodal Sampling. arXiv preprint arXiv:2503.03792.
43. Kannan Nova. "Machine Learning Approaches for Automated Mental Disorder Classification based on Social Media Textual Data". Contemporary Issues in Behavioral and Social Sciences, vol.7(1), 2023, pp. 70–83.
44. Karpathy and L. Fei-Fei, "Deep visual-semantic alignmentsfor generating image descriptions," in Proceedings of the IEEEComputer Society Conference on Computer Vision and PatternRecognition (CVPR), 2015, pp. 3128–3137
45. Khaled A., Neamat E.T., Ahmad H.H. Sentiment analysis over social networks: an overview. International conference in Systems, man and cybernetics: Proceedings, Hong Kong, 9-12 Oct. 2015. P. 2174–2179.

46. Konda Vaishnavi, U Nikhitha Kamath, B Ashwath Rao, N V Subba Reddy (2022). "Predicting Mental Health Illness using Machine Learning Algorithms". Journal of Physics: Conference Series, vol. 2161, 1st International Conference on Artificial Intelligence, Computational Electronics and Communication System (AICECS 2021) 28-30 October 2021, Manipal, India
47. Lu, W., Li, J., Li, Y., Sun, A., and Wang, J. (2020). A CNN-LSTM-based model to forecast stock prices. Complexity 2020:6622927. doi: 10.1155/2020/6622927
48. M. Oncescu, A. S. Koepke, J. F. Henriques, Z. Akata, andS. Albanie, "Audio Retrieval with Natural Language Queries,"in Proceedings of Conference of the International Speech Communication Association, 2021, pp. 2411–2415.
49. Mantyla M., Graziotin D., Kuutila M. The evolution of sentiment analysis — a review of research topics, venues and top cited paper. Computer science review. 2018. № 27. P. 16–32.
50. Mao, K.; Zhang, W.; Wang, D.B.; Li, A.; Jiao, R.; Zhu, Y.; Wu, B.; Zheng, T.; Qian, L.; Lyu, W.; et al. Prediction of depression severity based on the prosodic and semantic features with bidirectional LSTM and time distributed CNN. IEEE Trans. Affect. Comput. 2022, 14, 2251–2265. https://doi.org/10.1109/taffc.2022.3154332.
51. Mao, K.; Zhang, W.; Wang, D.B.; Li, A.; Jiao, R.; Zhu, Y.; Wu, B.; Zheng, T.; Qian, L.; Lyu, W.; et al. Prediction of depression severity based on the prosodic and semantic features with bidirectional LSTM and time distributed CNN. IEEE Trans. Affect. Comput. 2022, 14, 2251–2265.
52. Miao, X.; Li, Y.; Wen, M.; Liu, Y.; Julian, I.N.; Guo, H. Fusing features of speech for depression classification based on higher-order spectral analysis. Speech Commun. 2022, 143, 46–56.]
53. Miao, X.; Li, Y.; Wen, M.; Liu, Y.; Julian, I.N.; Guo, H. Fusing features of speech for depression classification based on higher-order spectral analysis. Speech Commun. 2022, 143, 46–56. https://doi.org/10.1016/j.specom.2022.07.006.
54. Michael M. Tadesse; Hongfei Lin; Bo Xu; Liang Yang. "Detection of Depression-Related Posts in Reddit Social Media Forum Publisher". IEEE Access, vol.7,

2019, pp. 44883 – 44893.

55. Mienye, Ibomoiye Domor, Theo G. Swart, and George Obaido. 2024. "Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications" Information 15, no. 9: 517. https://doi.org/10.3390/info15090517
56. Milintsevich, K.; Sirts, K.; Dias, G. Towards automatic text-based estimation of depression through symptom prediction. Brain Inform. 2023, 10, 4.
57. Muhammad Arif, Iqra Ameer, Necva Bölücü, Grigori Sidorov, Alexander Gelbukh, Vinnayak Elangovan. "Mental Illness Classification on Social Media Texts using Deep Learning and Transfer Learning" . Computacion y Sistemas vol. 28(2), (2024), pp.451-464
58. N. Shakhovska, N. Boyko, P. Pukach. The Information Model of Cloud Data Warehouses International Conference on Computer Science and Information Technologies, CSIT 2018, September 11-14, Lviv, Ukraine, 2019, pp. 182-191.
59. Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and MingLiu, "Neural speech synthesis with transformer network," inProceedings of the AAAI Conference on Artificial Intelligence,2019, vol. 33, pp. 6706–6713.
60. Nandi, P. (2025). CNNs for Audio Classification - TDS Archive - Medium. Available at: https://medium.com/data-science/cnns-for-audio-classification-6244954665ab
61. Nataliya Shakhovska, et. al.: "The Developing of the System for Autimatic Audio to Text Conversion", IT&AS'2021: Symposium on Information Technologies and Applied Sciences, March 5–6, 2021, Bratislava, Slovak Republic.
62. Nerella S, Bandyopadhyay S, Zhang J, Contreras M, Bumin A, Silva B, Sena J, Shickel B, Bihorac A, Rashidi P. "Transformers in healthcare: a survey". arXiv. https://doi.org/10.48550/arXiv.2307.00067
63. P. Zdebskyi, V. Lytvyn,Y. Burov, and et. Intelligent system for semantically similar sentences identification and generation based on machine learning methods, CEUR Workshop Proceedings, 2020, pp. 317–346.
64. Panek D. Acoustic analysis assessment in speech pathology detection / D. Panek, A. Skalski, J. Gajda, R. Tadeusiewicz // International Journal of Applied

Mathematics and Computer Science. — 2015. — Vol. 25, No. 3. — P. 631–643.

65. Panek, D., Skalski, A., Gajda, J., and Tadeusiewicz, R. (2015). Acoustic analysis assessment in speech pathology detection. Int. J. Appl. Math. Comput. Sci. 25, 631–643. doi: 10.1515/amcs-2015-0046
66. Park, J.; Moon, N. Design and implementation of attention depression detection model based on multimodal analysis. Sustainability 2022, 14, 3569.
67. Park, J.; Moon, N. Design and implementation of attention depression detection model based on multimodal analysis. Sustainability 2022, 14, 3569. https://doi.org/10.3390/su14063569.
68. Pavaloaia V.D., Teodor E.M., Fotache D., Danilet M. Opinion mining on social media data sentiment analysis of user preferences. Sustainability. 2019. Vol. 11. № 4459. P. 1–21.
69. Rao, Chengping, and Yang Liu. "Three-dimensional convolutional neural network (3D-CNN) for heterogeneous material homogenization." Computational Materials Science 184 (2020): 109850.
70. Rastogi, M. (2020) Tutorial on lstms: a computational perspective. Towards data science.
71. Roberts, L. Understanding the Mel Spectrogram. 2020. Available online: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53 (accessed on 22 December 2024).
72. S. Chowdhury and J. Sil, "FACERECOGNITION from NON-FRONTALIMAGES Using DEEP NEURALNETWORK," in 2017 Ninth InternationalConference on Advances in PatternRecognition (ICAPR), 2017, pp. 1-6.
73. Shahzad Qaiser, Ramsha Ali. “Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents”. School of Quantitative Sciences, International Journal of Computer Applications (0975 - 8887), vol. 181 - No.1, 2018, pp.25 - 29.
74. Shbair, Wazen M., Thibault Cholez, Jerome Francois, and Isabelle Chrisment. "A multi-level framework to identify HTTPS services." In NOMS 2016-2016

IEEE/IFIP Network Operations and Management Symposium, pp. 240-248. IEEE, 2016.

75. Sidhu, Manjit Singh, Nur Atiqah Abdul Latib, and Kirandeep Kaur Sidhu. "MFCC in audio signal processing for voice disorder: a review." Multimedia Tools and Applications (2024): 1-21.
76. T. Saravanan, T. Jhaideep, N. Hima Bindu. “Detecting depression using Hybrid models created using Google's BERT and Facebook's Fast Text Algorithms”. 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2022.
77. Tuncer T. Novel multi center and threshold ternary pattern based method for disease detection method using voice / T. Tuncer, S. Dogan, F. Ozyurt, [et al.] // IEEE Access. — 2020. — Vol. 8. — P. 84532–84540.
78. Vankayala Tejaswini, Korra Sathya Babu, Bibhudatta Sahoo. “Depression Detection from Social Media Text Analysis using Natural Language Processing Techniques and Hybrid Deep Learning Model”.
79. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need (arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762
80. Vitaly Yakovyna, Natalya Shakhovska, "Software failure time series prediction with RBF, GRNN, and LSTM neural networks", Procedia Computer Science 207(4):837-847, DOI:10.1016/j.procs.2022.09.139.
81. Warda Rahim. “Modelling Insurance Claims Data Using the Tweedie Approach”. Medium, May 15, 2022. URL: https://medium.com/@wardarahim25/modelling-insurance-claims-data-using-the-tweedie-approach-94db8b14bfb5. (дата звернення: 20.11.2024).
82. Wisam Abdulazeez Qader, Musa M.Ameen, Bilal I. Ahmed. “An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges”. Conference: 2019 International Engineering Conference (IEC).
83. Yang, J.; Luo, F.-L.; Nehorai, A. Spectral contrast enhancement: Algorithms and comparisons. Speech Commun. 2003, 39, 33–46.

84. Yin, F.; Du, J.; Xu, X.; Zhao, L. Depression detection in speech using transformer and parallel convolutional neural networks. Electronics 2023, 12, 328.
85. Zhao, Y.; Liang, Z.; Du, J.; Zhang, L.; Liu, C.; Zhao, L. Multi-head attention-based long short-term memory for depression detection from speech. Front. Neurorobotics 2021, 15, 684037.
86. Zheliznyak, Z. Rybchak, I. Zavuschak, Analysis of clustering algorithms, 2017. Advances in Intelligent Systems and Computing, 2017, pp. 305–314.
87. Zoryana Rybchak, et. al. "Analysis of methods and means of text mining". ECONTECHMOD, 6(2), 2017, pp. 73-78.
88. Басистюк О. А., Мельникова Н. І. Мультимодальне розпізнавання мовлення на основі звукових і текстових даних // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2022. – № 5 (313). – С. 22–25.
89. Бехта І. А., Матвієнків О. С. Структурно-семантичні типи фразеологізмів в англійськомовному художньому прозовому тексті. Вчені записки ТНУ імені В. І. Вернадського. Серія: Філологія. Соціальні комунікації, УДК 821.111-112. 81’42, DOI https://doi.org/10.32838/2663-6069/2020.2-2/05
90. Дарчук Н. Лінгвістичні засади автоматичного сентимент аналізу українськомовного тексту. Science and education a new dimension. 2019. № 189. С. 10–13.
91. Датасет “reddit-mental-illness-82”. URL: https://huggingface.co/datasets/mavinsao/reddit-mental-illness-82
92. Іванов Є.М., Коваленко С.В. Розробка web-додатка для аналізу тональності текстової інформації. Інформаційні технології: наука, техніка, технологія, освіта, здоров’я. 2017. Ч. 1. С. 18–19.
93. Немеш О., Романюк А., Теслюк В. Аналіз тональності тексту: основні поняття та приклади застосування. Людина, комп’ютер, комунікації: зб. тез доп. міжнар. наук.-практ. конф., м. Львів, квіт. 2015 р. Львів, 2015. С. 47–49.
94. О. Басистюк, С. Гимон. “Застосування машинного навчання в оцінці психоемоційних станів людини”. 14-та міжнародна науково-технічна

конференція: "Інформація, комунікація, суспільство – 2025"

95. Олег Басистюк, Наталія Мельникова, Ірина Думин, Андрій Думин. Ансамблевий підхід у мультимодальній обробці даних на основі Google API // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2024. – № 4 (339). – С. 235–238
96. Рузакова О.В. Використання апаратів штучного інтелекту для формалізації фінансових об'єктів при побудові СППР / О. В. Рузакова, Н. П. Юрчук // Вісник Хмельницького національного університету: Технічні науки. – 2021. № 1. – С. 45-511.
97. Тональний словник української мови // GitHub. URl: https://github.com/lang-uk/tone-dict-uk
98. Шаховська Н.Б., Гірак Х.Ю. Шкалювання емоційно забарвлених слів для використання у методах класифікації тональності. Вісник Національного університету «Львівська політехніка». 2017. № 872(1). С. 195–203.
99. Шингалов Д.А., Мелешко Е.В., Минайленко Р.Н., Резниченко В.А. Методи автоматичного аналізу тональності контенту у соціальних мережах для виявлення інформаційно- психологічних впливів. Центральноукраїнський науковий вісник. 2017. № 30. С. 196–202.
100. Ялова К., Яшина К., Говорущенко Т., Тарасюк О. Сентимент аналіз засобами нейронної мережі. Математичне моделювання. 2021. № 1 (44). С. 30–37.

SCIENTIFIC EDITION

MONOGRAPH

ENTWICKLUNG DES WISSENSCHAFTLICHEN DENKENS
METHODEN UND WERKZEUGE ZUR INTELLIGENTEN ANALYSE UKRAINISCHSPRACHIGER TEXTE ZUR IDENTIFIZIERUNG EMOTIONALER MARKER UND ZUR BESTIMMUNG DER BETONUNG IM TEXT

SCIENTIFIC THOUGHT DEVELOPMENT

METHODS AND TOOLS FOR INTELLIGENT ANALYSIS OF UKRAINIAN-LANGUAGE TEXTS TO IDENTIFY EMOTIONAL MARKERS AND DETERMINE STRESS IN THE TEXT

MONOGRAPHIC SERIES «EUROPEAN SCIENCE»

BOOK 41. PART 2

Authors:
Dumyn I., Basystiuk O., Hymon S.

The scientific achievements of the authors of the monograph were also reviewed and recommended for publication at the international scientific symposium
«Entwicklung des wissenschaftlichen Denkens /
Scientific thought development '2025»
(June 30, 2025)

Monograph published in the author's edition

The monograph is included in
International scientometric databases

500 copies
June, 2025

Published:
ScientificWorld -NetAkhatAV
Lußstr 13,
Karlsruhe, Germany

e-mail: editor@promonograph.org
https://desymp.promonograph.org

ISBN 978-3-989240-98-8

https://desymp.promonograph.org

e-mail: editor@promonograph.org