Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2026.076083

ARTICLE

ComAlign: A Benchmark Aligning Natural Language with Operating System Commands

Shasha Li, Bin Ji*, Xiaodong Liu, Jun Ma and Jie Yu*

College of Computer Science and Technology, National University of Defense Technology, Changsha, China

*Corresponding Authors: Bin Ji. Email: jibin@nudt.edu.cn; Jie Yu. Email: yj@nudt.edu.cn

Received: 13 November 2025; Accepted: 29 December 2025; Published: 12 March 2026

ABSTRACT: Aligning natural language with operating system (OS) commands allows users to perform complex computer tasks through simple natural language descriptions. However, due to the complex nature of natural language, it still remains challenging to achieve precise alignment. In this paper, we present **ComAlign**, a Chinese benchmark dataset that pairs Chinese natural language descriptions with corresponding OS commands. ComAlign covers a broad range of 82 distinct OS command types with a total of 1811 natural language descriptions. We elaborate on the construction of ComAlign and construct three baselines to evaluate the alignment accuracy on ComAlign. Experimental results show that even advanced large language models struggle with certain ambiguously phrased OS commands. Specifically, the best performing baseline achieves 46.9% alignment accuracy. We demonstrate that ComAlign is collected from real-world application scenarios, making it particularly suitable for developing and benchmarking intelligent OS and agent systems that support user-machine interactions through natural language.

KEYWORDS: Large language model; language-command alignment; artificial intelligence operating system

1 Introduction

The ability for users to control computers through natural language is a long-standing goal of human-computer interaction [1]. And with the rapid development of artificial intelligence (AI) assistants and AI personal computers (AIPC), aligning natural language with underlying os commands has become an increasingly important research direction. Modern AI assistants aim to execute tasks such as adjusting system settings, retrieving information, and managing schedules based on spoken or written descriptions [2,3]. With the advent of large language models (LLMs), there is renewed potential for such AI assistants to understand complex natural language instructions [4,5]. However, a crucial challenge remains: how to precisely align a natural language user description with its corresponding operating system (OS) command [6,7]. For example, if a user says *"I forgot my password, I need to set a new one"*, the system must recognize this as a request to invoke the OS function for changing the user password. Misaligning the user's intent with an incorrect command could lead to failure or unintended side effects. To facilitate studies on this task, Lin et al. [8] introduce the NL2Bash dataset for aligning English natural language to Linux commands, which contains approximately 10,000 commonly used one-line Bash commands paired with their English natural language descriptions. Preliminary studies demonstrate the considerable difficulty of this task. Specifically, even when restricted to single-line commands, the baselines achieve only about 36% Top-1 alignment accuracy. The NLC2CMD [9,10] competition organized at NeurIPS further advanced this research area, improving alignment accuracy to 53.2% through the introduction of Transformer-based models and other

advanced techniques. However, existing studies and datasets are predominantly focused on English, while the alignment between natural language and OS commands in Chinese remains largely unexplored and under-resourced [11]. Due to the distinct linguistic characteristics of Chinese, such as differences in lexical structure and semantic ambiguity, directly applying approaches trained on English data often yields suboptimal performance. Moreover, we'd like to emphasize that Chinese text processing presents unique linguistic challenges due to the absence of explicit word boundaries, flexible word formation, and high semantic ambiguity. For example, a short character sequence may correspond to multiple valid segmentations or meanings depending on context. These characteristics increase the complexity of modeling Chinese texts.

To tackle the above issue, we introduce **ComAlign**, a Chinese benchmark dataset designed to systematically support exploring approaches to align Chinese natural language descriptions with the corresponding OS commands. Our primary contribution is a curated dataset of diverse natural language user instructions in Chinese paired with their aligned OS command descriptions. ComAlign includes a wide spectrum of operation tasks that typical users might request, ranging from system operations (e.g., changing settings or managing hardware devices) to question & answer tasks. In total, it encompasses 82 distinct OS command types and 1811 natural language descriptions, making it, as far as we know, the first comprehensive Chinese benchmark dataset for this language-command alignment task. The dataset is intended to facilitate training and evaluation of models that act as an interface between the user's natural language descriptions and OS commands.

We formulate the language-command alignment task primarily as an intent classification problem, i.e., given a user's natural language description, predict which OS command (from the predefined set) is being requested, and identify any necessary parameters. For example, a natural language description like "屏幕太暗了，我想调高下亮度" (the screen is too dark, I want to increase the brightness) should be mapped to the "设置屏幕亮度" (Set Screen Brightness) command with an appropriate parameter, e.g., "调高" (increase). To establish baseline performance on ComAlign, we design and implement three benchmark baselines, including a vector matching-based baseline (VM), an input normalisation and vector matching-based baseline (INVM), and a vector matching with LLM-powered intent-consistency evaluation-based baseline (VMICE). We conduct comprehensive analyses of their performance on the ComAlign dataset. Experimental results show that even the LLM-powered VMICE baseline solely achieves 46.9% alignment accuracy on ComAlign, highlighting the inherent difficulty of this task and underscoring the need for substantial further study to advance this task.

Compared with existing text-to-SQL and command-line benchmarks, constructing a Chinese natural language–OS command dataset presents several unique challenges. First, OS commands often correspond to multi-step Graphical User Interface (GUI) operations, so it is non-trivial to abstract them into concise, reusable command templates. Second, Chinese user instructions in daily computer use are highly colloquial and elliptical. For example, users tend to simply describe a state (e.g., "屏幕太暗了", "the screen is too dark") instead of explicitly specifying the desired operation. Third, there exists a strong many-to-one relationship between natural language descriptions and OS commands. These properties motivate the design of our ComAlign benchmark.

In summary, our contributions can be summarized as follows:

(1) **Construct a Benchmark Dataset.** We collect 1811 pairs of <natural language description, operating system command> to construct the ComAlign dataset. The dataset covers 82 types of OS commands. To the best of our knowledge, this is the first dataset in the Chinese language domain for the purpose of aligning natural language with operating system commands.

(2) **Design Reproducible Baselines.** For the task of aligning natural language with OS commands, we design three reproducible baselines. These baselines leverage fastText and LLMs to perform vector

matching, input normalization, and intent-consistency evaluation, providing strong and reproducible baselines to facilitate future studies.

(3) **Extensive Experiments and Analysis.** We conduct comprehensive experiments and analyses of the three baselines on the ComAlign dataset. Experimental results highlight the intrinsic difficulty of aligning natural language with OS commands and provide insights that point toward promising directions for future studies.

2 Related Work

2.1 Aligning Natural Language with OS Commands

Early studies formalize the task of aligning natural language with OS commands as a semantic parsing task [12]. A representative early work in this area is the NL2Bash dataset. Lin et al. [8] collect a set of commonly used Linux Bash commands and their corresponding English descriptions from technical question & answering forums, tutorials, and other online sources. After quality control and filtering, they released 10,000 text–command pairs covering over 100 Bash utilities. They also proposed several baselines, including Seq2Seq, CopyNet, and a rule-based heuristic baseline named TELLINA, establishing an initial performance benchmark for this task. Subsequent studies further expanded this study. The NLC2CMD [9,10] competition organized at NeurIPS introduced a larger-scale English NL2Bash dataset and new evaluation metrics. Transformer-based models and other advanced techniques are proposed, substantially improving alignment accuracy to 53% on full-command prediction. Fu et al. [13] conduct in-depth analyses of top-performing approaches, summarize optimization strategies, and propose an improved workflow that achieves new state-of-the-art performance on the NLC2CMD dataset.

2.2 Aligning Natural Language with Codes

Beyond OS commands, aligning natural language with code (NL2Code) [14] has attracted growing attention in recent years. On one hand, extensive work has focused on automatic generation and understanding of general programming codes [15]. For example, Microsoft introduced the CodeXGLUE benchmark [16], which encompasses ten types of tasks, including code clone detection, code search, code summarization, and text-to-code generation, across multiple programming and formal languages (e.g., SQL). These tasks essentially aim to align natural language to specific code semantics and can be framed as program synthesis problems, i.e., generating code that satisfies given specifications such as natural language descriptions or input–output examples. With the advancement of deep learning and pretrained language models, many powerful code generation models have emerged. Transformer-based pretrained models such as CodeGPT [16], CodeBERT [17], and CodeLLaMA [18] have established strong baselines across CodeXGLUE tasks. More recently, large-scale models like OpenAI Codex [19] and DeepMind AlphaCode [20] have achieved near-human-level performance on general code generation benchmarks. On the other hand, research on NL2Code in non-English contexts lags behind English contexts [15]. Due to the lack of large-scale annotated datasets, most prior work on text-to-code generation has focused on English. To promote progress in the Chinese domain, the community has begun constructing corresponding datasets and benchmarks. For example, Wang et al. [11] release DuSQL, a Chinese cross-domain text-to-SQL dataset containing 23,797 pairs of Chinese questions and corresponding SQL queries. This work fills a crucial data gap for semantic parsing in Chinese database queries.

3 Dataset Construction

3.1 Data Collection and Annotation

Source and Scenarios. Our dataset originates from real personal-computer usage scenarios, primarily using an open-source desktop OS (i.e., openKylin) and its common applications. We simulate typical user environments and compile a list of system functions covering system settings, application launch, multimedia control, network management, and so on.

Annotation Process. We first enumerate potential OS operations, e.g., “修改用户密码” (modify user password), “蓝牙设置” (bluetooth setting), “设置屏幕亮度” (set screen brightness). Then, we draft a normalized OS command description serving as the unique identifier for each type of operation. Next, we collect various Chinese natural language expressions to describe these operations, including colloquial, formal, and incomplete phrases. These expressions are carefully checked and bound to the corresponding OS command, with parameters (e.g., volume level, brightness percentage) recorded as needed. It should be noted that the dataset is collected directly from real-world application scenarios rather than being manually crafted or annotated by human annotators. As a result, there is no explicit annotation process involving multiple annotators, and issues such as inter-annotator agreement or annotation consistency procedures do not apply in this context. Each data entry reflects naturally occurring system records generated during actual usage, ensuring authenticity and practical relevance of the dataset.

The resulting dataset contains 1811 pairs of natural language descriptions and corresponding normalized OS command descriptions. Among them are 1706 OS operation tasks and 105 question & answering tasks. Each data entry includes the task type, normalized OS command description, natural language description, and optional parameters. We identify 82 types of OS commands, where each type has one normalized command description and 18 to 25 natural language descriptions. The majority (81.61%) of these natural language descriptions contain 6 to 14 Chinese tokens. However, we also add both short and long natural language descriptions to increase the diversity. For example, the shortest natural language description has 3 Chinese tokens while the longest contains 28 Chinese tokens. We present the detailed Chinese token counts of the natural language descriptions in Fig. 1.

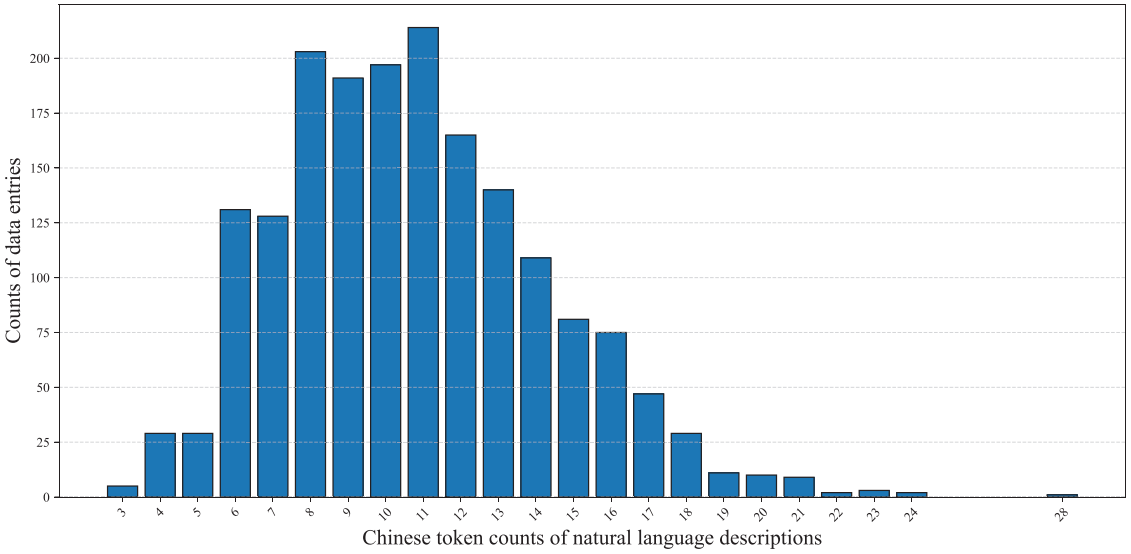
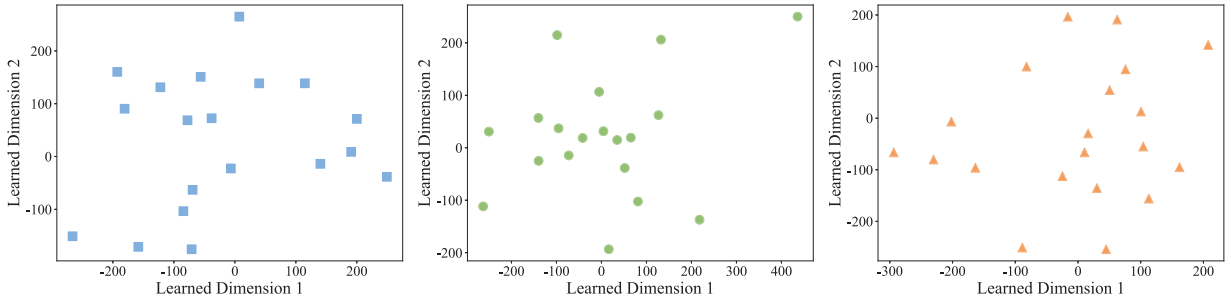


Figure 1: Chinese token count statistics of natural language descriptions

We visualize the semantic representation of natural language descriptions to analyze the semantic distribution of natural language descriptions across different normalized OS command descriptions. Specifically, the selected normalized OS command descriptions are “修改用户密码” (modify user password), “设置屏幕亮度” (set screen brightness), “蓝牙设置” (bluetooth setting). For each normalized OS command description, we first transformed its natural language descriptions into numerical vectors using the Term Frequency–Inverse Document Frequency (TF–IDF) representation, and subsequently applied dimensionality reduction techniques (PCA and t-SNE) for visualization. Fig. 2 presents the visualization results, from which we can observe that the natural language descriptions are well distributed in the embedding space, indicating the diversity of these descriptions.



(a) Modify user password **(b)** Set screen brightness **(c)** Bluetooth setting

Figure 2: The visualization results of semantic distributions of natural language descriptions across three types of OS commands

Additionally, we’d like to emphasize that natural language descriptions and normalized OS command descriptions form a many-to-one relationship. Fig. 3 takes the normalized OS command “修改用户密码” (modify user password) as an example and presents some natural language descriptions that should align with it.

We report more analyses of the ComAlign dataset in Section 6.1.

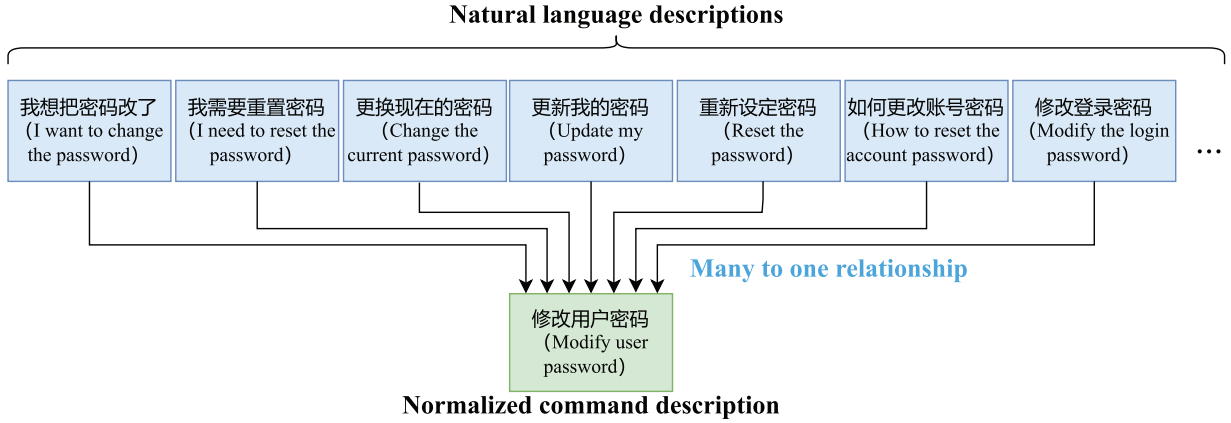


Figure 3: A case study of the many-to-one relationship between the natural language descriptions and the normalized command description

3.2 Task-Type Distribution and Parameters

To analyse the ComAlign dataset further, we count task types and parameter distributions:

(1) OS operations vs. question-answering. Approximately 94.2% (1706) of data entries involve OS operations, and 5.8% (115) are question & answering tasks. The latter task requires invoking a large language model to generate answers and thus are excluded from alignment accurracy evaluation.

(2) Parameterised vs. non-parameterised tasks. Among the 1706 natural language descriptions of OS operations, around 868 are parameterised (e.g., "volume control", "set screen brightness"), while the remaining 838 have no parameters (e.g., "modify user password"). Parameterised descriptions span 43 types of OS commands and non-parameterized ones 39 types of OS commands. Parameterised descriptions often involve numeric or directional adjustments, whereas non-parameter descriptions tend to be simple toggles or navigation actions.

We report more task-type and parameter distribution analyses in Section 6.1.2.

3.3 Multi-Parameter Instances

In our current dataset scheme, each data entry in ComAlign is associated with a single canonical intent category, and linguistically complex user commands that mention several sub-goals are normalized to one OS-level operation whenever they can be executed as a single command. The remaining complexity is mainly reflected in the number of explicit slot arguments attached to each intent. Concretely, among all the data entries, 76 examples (4.2%) involve two explicit slots (for example, specifying both an operation type and a time-zone name in "添加东京时区"). If we regard these two-slot cases as "multi-parameter" or "multi-intent-like" commands in the sense that the model must satisfy multiple constraints simultaneously, then they account for 4.2% of the ComAlign dataset.

4 Baseline Construction

This section outlines three baselines for aligning natural language descriptions with normalized OS command descriptions, which are named as Vector Matching-based baseline, Input Normalisation and Vector Matching-based baseline, and Vector Matching with Intent-Consistency Evaluation-based baseline.

4.1 Vector Matching-Based Baseline (VM)

The first baseline performs alignment by nearest-neighbour search in a embedding space. Each natural language description in the dataset is segmented and mapped to 300-dimensional fastText embeddings. The vector representation of a description is the mean of its word vectors. The vectors form a database for fast nearest-neighbour descriptions. For a user input, we compute its mean vector and retrieve the database vector with the best similarity. The normalized OS command description corresponding to this most similar vector is returned as the candidate. To avoid mismatches, we apply a similarity threshold (initially set to 0.90): if the highest similarity falls below this threshold, the input is treated as a question & answering task. Fig. 4 illustrates the architecture of the VM baseline. The VM baseline is efficient and does not require a large language model, but fastText embeddings cannot capture complex syntax and context. Consequently, colloquial or long sentences are prone to errors.

In this study, we explore calculating the similarity using the Cosine similarity and Euclidean distance. We report the exploration results in Section 5.

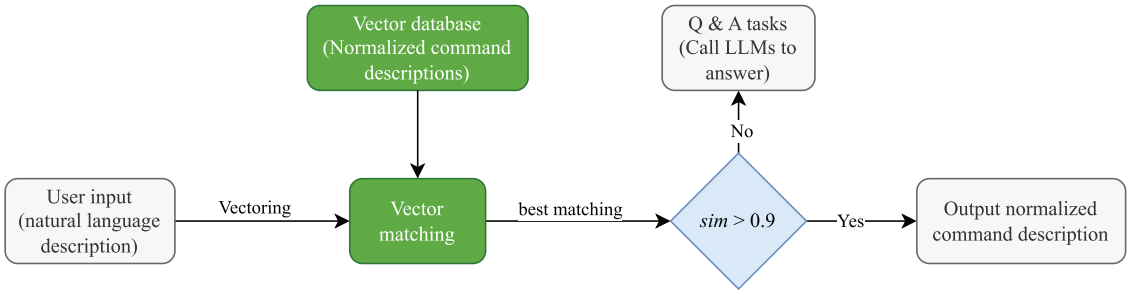


Figure 4: The architecture of the vector matching-based baseline

4.2 Input Normalisation and Vector Matching-Based Baseline (INVM)

The second baseline adds a user input normalisation step before vector matching to reduce the impact of colloquialisms and redundant words. We design a prompt with examples such as "User input: please help me turn on Bluetooth; Normalised input: turn on Bluetooth", instructing an LLM to rewrite any user input into a concise command phrase. When a new input arrives, we generate a normalised input via an LLM, preserving the core action and parameters while removing pleasantries or background. We then embed this normalised input with fastText and perform vector matching with the same algorithm presented in Section 4.1. Fig. 5 illustrates the architecture of the INVM baseline. Because the normalisation step eliminates many colloquial words and modifiers, this method improves matching accuracy by making expressions more uniform. However, we also observe that the normalization step causes performance drops in some scenarios. We present detailed analyses in Section 5.

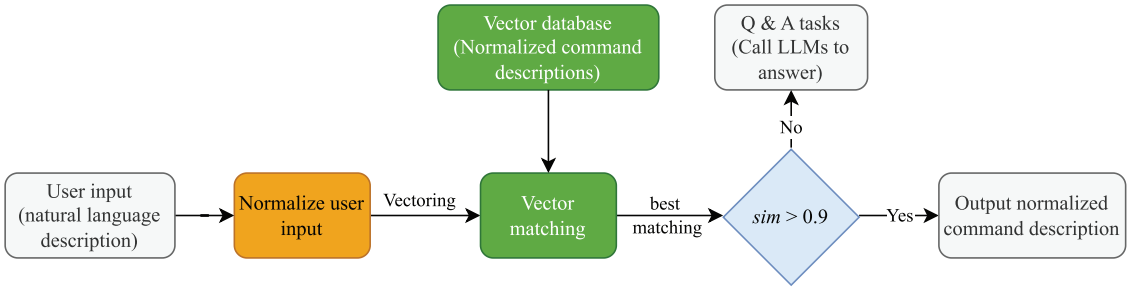


Figure 5: The architecture of the input normalisation and vector matching-based baseline

4.3 Vector Matching with Intent-Consistency Evaluation-Based Baseline (VMICE)

The third baseline introduces an *intent-consistency evaluation* using an LLM to overcome the limitations of a fixed similarity threshold. We first apply the vector matching as aforementioned to obtain the candidate description. Then we construct a question for an LLM, i.e., "*Please judge whether the following two descriptions refer to the same system operation. Answer only 'Yes' or 'No'. description 1: normalized input, description 2: best-matching natural language description.*" If the LLM outputs "*Yes*", we return the OS command associated with the best-matching natural language description; otherwise, we treat the normalized user input as a question & answering task. The evaluation prompts include examples and highlight action verbs and parameters. This VMICE baseline does not rely on a fixed threshold and can handle semantically equivalent sentences with low vector similarity. Fig. 6 illustrates the architecture of the VMICE baseline. In Section 5, we present detailed experimental results and analyses.

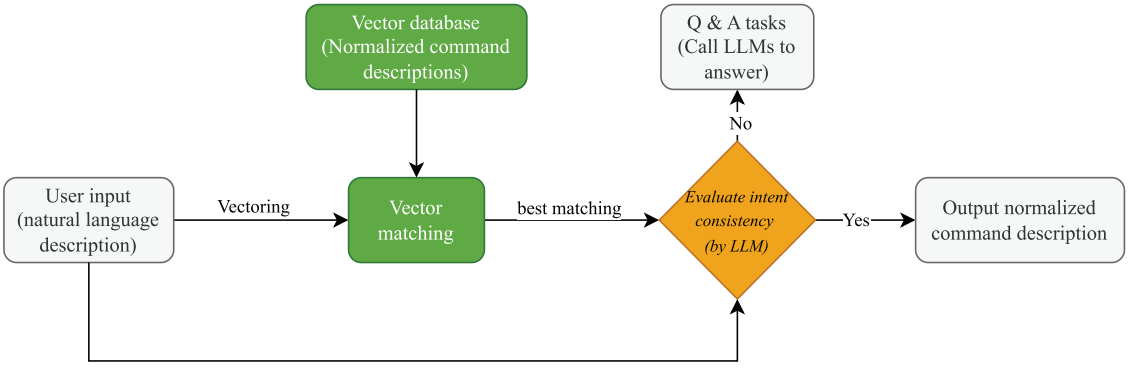


Figure 6: The architecture of the vector matching and intent-consistency evaluation-based baseline

4.4 Future Extensions

While this work focuses on lightweight and task-specific baselines to highlight the inherent challenges of the problem, future research could explore the integration of contextual embedding models and modern Chinese sentence encoders. For instance, large language models, such as Qwen [21,22], LLaMA [23,24], and GPT [25] series models, potentially further improve performance on complex or ambiguous cases. We leave a systematic investigation of these advanced models for future work.

5 Experiments

5.1 Experimental Settings

All experimental results are obtained by evaluating the proposed ComAlign dataset, and the evaluation metric is the *alignment accuracy*, which is the proportion of natural language descriptions that are aligned to the correct normalized OS command descriptions. We explore normalizing user inputs and intent-consistency evaluation with seven LLMs, i.e., FM9G-4B, FM9G-8B, FM9G-7B, (https://www.osredm.com/jiuyuan/CPM-9G-8B), miniCPM-4B [26], LLaMA-3.1-8B (LLaMA-8B for short) [24], Qwen2.5-7B (Qwen-7B for short) [21], and LLaMA-3.2-3B-Instruct (LLaMA-3B for short) [24]. The other experimental settings are shown below:

(1) **Embedding and matching.** Each natural language description is embedded into 300-dimensional fastText [27] vectors, and a vector database is built. Euclidean distance and Cosine similarity are used to calculate the similarity, respectively.

(2) **Input normalisation.** For the INVM baseline, we employ an LLM to normalise user input, guided by prompts emphasising concise commands. After normalisation, embedding and matching follow as the VM baseline. All seven LLMs are investigated to normalize user inputs.

(3) **Intent judgement.** For the VMICE baseline, the best-matching natural language description and the user input are fed to an LLM to evaluate their intent consistency. If the LLM responds “Yes”, the corresponding normalized OS command description is returned; otherwise, an LLM is called to respond to the user input.

To fairly evaluate the three baselines on the proposed ComAlign benchmark dataset, we adopt leave-one-out cross-validation. Each time, one instance from the 1811 pairs is held out for testing and the rest serve as training data. Only OS operations are included in the alignment accuracy calculation.

5.2 Main Results

Table 1 summarises the alignment accuracy of the three baselines under four similarity thresholds (0.9, 0.8, 0.7 and 0.6). The results reveal several notable patterns and provide insights into the linguistic and technical challenges inherent to the ComAlign dataset.

(1) **Performance comparison across baselines.** Among the three baselines, the VMICE baseline consistently achieves the highest alignment accuracy across all threshold settings, followed by the INVM baseline, while the VM baseline performs the worst. For instance, at a threshold of 0.9 using Euclidean distance, the average alignment accuracies of the VMICE, INVM, and VM baselines are 43.6%, 40.7%, and 39.2%, respectively. These trends confirm the benefit of incorporating LLMs in both the normalization and intent-consistency evaluation phases. The INVM baseline improves upon VM by reducing noise from redundant or colloquial expressions through input normalization, enabling more semantically compact and comparable representations. Meanwhile, the VMICE baseline introduces a secondary intent-consistency check that allows the baseline to correctly identify semantically equivalent descriptions even when their surface forms differ significantly. This step effectively compensates for the limitations of fastText embeddings, which primarily rely on shallow lexical similarity. The consistent performance gain of VMICE suggests that ComAlign contains a substantial proportion of paraphrased or semantically ambiguous user expressions that simple vector similarity cannot fully capture.

(2) **Influence of similarity thresholds.** For all the three baselines, higher similarity thresholds lead to better performance. Taking the VM baseline as an example, when the threshold is increased from 0.6 to 0.9, alignment accuracy improves from 32.7% to 39.2%. A similar trend is observed across INVM and VMICE. This pattern implies that the semantic representations produced by fastText embeddings sometimes overestimate similarity between unrelated descriptions. Higher thresholds act as a filter, excluding spurious matches caused by shared context words (e.g., “设置” setting or “打开” open) that frequently appear in multiple commands but do not necessarily indicate the same intent. Therefore, stricter thresholds enhance precision by ensuring that only those pairs with stronger semantic overlap are accepted as aligned commands.

(3) **Impact of similarity metrics.** Comparing Cosine similarity and Euclidean distance, we find that Euclidean distance yields slightly better results in 9 out of 12 setting scenarios, though the overall difference is modest (typically below 0.5%). This observation suggests that both metrics capture comparable relational structures in the embedding space. The minor advantage of Euclidean distance may stem from its higher sensitivity to absolute vector magnitudes, which can help discriminate semantically close but non-identical expressions in the fastText embedding space.

(4) **Performance against task type and linguistic variation.** Further examination shows that alignment accuracy varies across task categories. Commands with clear, distinct semantics (e.g., “修改用户密码” (modify user password), “打开蓝牙”) (open Bluetooth) are matched more accurately, while tasks involving graded or parameterized operations (e.g., brightness or volume adjustment) exhibit lower accuracy. This reflects the inherent ambiguity of parameterized instructions in natural language, where modifiers such as “调高一点;” (a bit higher) or “稍微亮一点” (a bit brighter) can introduce uncertainty in both intent and parameter magnitude. Additionally, commands expressed in colloquial or mixed-form language (e.g., “把声音开大点儿吧”, turn up the volume) tend to confuse purely vector-based approaches, underscoring the linguistic diversity of ComAlign and highlighting why normalization and intent-consistency alignment steps substantially improve performance.

(5) **Model stability and LLM effects.** When comparing across the seven LLMs used in INVM and VMICE, we observe modest variability (typically within ±3%) in alignment accuracy, indicating

stable behavior across model architectures and sizes. Larger instruction-tuned models (e.g., Qwen-7B and LLaMA-8B) generally outperform smaller ones, suggesting that an LLM's instruction-following and semantic generalization capabilities contribute directly to improved normalization and intent-consistency evaluation. Notably, FM9G-8B achieves the best overall score (46.9%) under VMICE, implying that LLMs trained on Chinese-heavy corpora may capture linguistic subtleties in user expressions better than multilingual or English-dominant LLMs.

Table 1: Alignment accuracy of the three baselines on the ComAlign benchmark dataset. "Euclidean" denotes calculating the similarity with the Euclidean distance. Similarly, "Cosine" denotes using the Cosine similarity to calculate the similarity. "Avg." denotes the average alignment accuracy score across the seven LLMs. Note that the VM baseline doesn't include an LLM in its implementation

Baseline	Similarity		LLM							Avg. (%)
	Algorithm	Threshold	FM9G-4B (%)	FM9G-7B (%)	FM9G-8B (%)	LLaMA-3.2-3B (%)	MiniCPM-4B (%)	Qwen2.5-7B (%)	LLaMA-3.1-8B (%)	
VM	Euclidean	0.9	–	–	–	–	–	–	–	39.2
	Cosine		–	–	–	–	–	–	–	38.9
	Euclidean	0.8	–	–	–	–	–	–	–	37.4
	Cosine		–	–	–	–	–	–	–	35.6
	Euclidean	0.7	–	–	–	–	–	–	–	33.2
	Cosine		–	–	–	–	–	–	–	34.1
	Euclidean	0.6	–	–	–	–	–	–	–	32.7
	Cosine		–	–	–	–	–	–	–	33.4
INVM	Euclidean	0.9	39.3	38.1	37.9	40.1	42.4	42.7	44.6	40.7
	Cosine		40.4	37.5	38.6	41.2	40.5	43.2	43.0	40.6
	Euclidean	0.8	36.7	37.4	38.2	38.1	37.6	40.1	41.1	38.5
	Cosine		35.8	37.3	38.9	37.2	38.5	39.2	41.6	38.4
	Euclidean	0.7	34.6	37.1	36.8	36.6	38.4	39.2	39.5	37.5
	Cosine		35.3	36.7	34.2	36.8	37.2	38.5	41.0	37.1
	Euclidean	0.6	33.0	35.8	34.6	35.2	36.7	38.2	37.6	35.9
	Cosine		34.6	34.9	34.2	35.5	36.0	37.2	36.4	35.5
VMICE	Euclidean	–	44.9	39.3	46.9	39.6	44.2	44.2	46.4	43.6
	Cosine		44.2	38.7	45.6	40.2	43.8	44.0	46.5	43.3

In summary, the above results reveal that the simple VM baseline is insufficient for robust natural language to OS command alignment due to its inability to handle paraphrasing and context ambiguity. The INVM baseline mitigates surface-level variance, while the VMICE baseline further bridges the semantic gap between diverse user expressions and canonical OS commands. Nonetheless, even the best-performing baseline achieves only 43.6% average alignment accuracy, underscoring the complexity of the ComAlign dataset and the substantial room for improvement. Future work should explore richer embedding models, contextual parameter extraction, and hybrid retrieval-generation architectures to further advance alignment performance.

5.3 Comparison of Baselines

Compared to the VM baseline, the INVM baseline includes a component of input normalization, and the VMICE baseline leverages LLMs to evaluate the intent-consistency of the user input and the best-matching natural language description. By analyzing the results reported in Table 1, we draw the following conclusions:

(1) **Limitations of vector matching.** Using fastText embeddings combined with Euclidean distance yields only 39.2% alignment accuracy on the ComAlign dataset when setting the threshold to 0.9. This relatively low performance reflects fundamental limitations in relying purely on word embeddings and vector-space similarity for semantic alignment. First, fastText, although efficient, lacks deep

contextual understanding. It treats words largely in isolation and cannot effectively handle polysemy or phrasal nuance. As a result, it struggles to distinguish between closely related instructions, such as “设置屏幕超时时间” (set screen timeout) and “设置睡眠时间” (set sleep time), which require understanding subtle intent differences. Second, the method is highly sensitive to surface-level features such as colloquial expressions, function words, and word order. For instance, users may phrase the same instruction in several informal ways, e.g., “我要关蓝牙” vs. “蓝牙关一下”, but these may diverge in fastText space due to token variation. Moreover, the method does not learn to ignore semantically irrelevant stopwords, causing it to overvalue irrelevant token overlap. Finally, the approach is particularly brittle in low-resource conditions, where training data are sparse or unevenly distributed across instruction types. Without fine-tuned supervision or task-specific adaptation, vector similarity alone fails to generalize well across description patterns.

(2) **Moderate but necessary improvement from normalisation.** Introducing a pre-processing step that normalizes user inputs, i.e., through token standardization, stopword removal, synonym replacement, and punctuation cleanup, yields a modest yet consistent improvement in accuracy, raising it by over 1.0% on average alignment accuracy. For the simple VM baseline, this is a non-trivial gain, especially considering that LLMs like miniCPM-4B, Qwen-7B, and LLaMA-8B achieve 42.1%, 42.7%, and 44.6% alignment accuracy after post-normalisation. The improvement stems from reducing the lexical variability that fastText struggles with. By eliminating surface noise and regularizing key terms (e.g., converting “把屏幕亮点” to a common form, i.e., “调高亮度”), normalization increases the chance that semantically similar descriptions will be matched in embedding space. In effect, it reduces the embedding dispersion caused by informal phrasing or user-specific expression. However, the relatively small gain also reveals a ceiling on what normalization can do. While it helps bridge superficial lexical gaps, it does not address deeper semantic ambiguities or compositional reasoning challenges. Thus, although normalization is essential for robustness in low-resource settings, it is not sufficient to resolve the alignment task comprehensively.

(3) **Notable gain from intent-consistency evaluation.** Substituting the fixed similarity threshold in the VM baseline with LLM-based intent-consistency evaluation yields a more substantial improvement, with an average accuracy increase of approximately 4.0% average alignment accuracy scores, and up to +7.7% scores compared to the VM baseline. This gain reflects the strength of LLMs in semantic inference and pragmatic understanding. Unlike fixed similarity thresholds, LLMs can reason about paraphrases, handle implicit intent, and disambiguate vague expressions. For example, given a query like “我不想让电脑太快休眠” (“I don’t want the computer to sleep too soon”), an LLM can correctly infer the intent to change the sleep timeout, even if that phrase is absent. Moreover, LLM-based intent-consistency evaluation adapts dynamically to different description contexts, rather than applying a uniform similarity threshold across all inputs. This flexibility enables the VMICE baseline to better distinguish near-miss cases that would otherwise fall on the wrong side of a similarity cutoff. The gains are particularly evident in categories with high lexical variance (e.g., display settings, connectivity options), where traditional methods falter due to surface mismatch. The results suggest that incorporating semantic decision-making is critical for advancing real-world command alignment systems.

6 Analyses

This section provides an in-depth examination of the ComAlign dataset and the performance behavior of the three proposed baselines from multiple perspectives. We analyze statistical properties, linguistic and semantic diversity, and baseline characteristics, and finally provide a qualitative discussion of common failure patterns and underlying challenges.

6.1 Dataset Analysis

6.1.1 Dataset Composition and Structural Characteristics

To the best of our knowledge, the ComAlign dataset is the first Chinese benchmark that systematically aligns natural language with OS commands. It contains 1811 manually curated pairs of natural language descriptions and corresponding normalized OS command descriptions. These pairs are distributed across 82 distinct command types, covering system configuration, application control, device management, network operations, etc. Each command type is associated with an average of 22 Chinese token expressions, forming a many-to-one mapping between natural language descriptions and canonical normalized OS command descriptions. A fine-grained inspection reveals two primary categories of tasks, as shown below:

(1) OS operation tasks (94.2%), which require mapping a natural language description to a normalized OS command description.
(2) Question-answering tasks (5.8%), which call for invoking an LLM to respond to the task.

The clear separation between these categories allows us to evaluate alignment accuracy independently. Within the OS operation subset, roughly half of the commands are parameterized, i.e., they require arguments such as brightness level, volume percentage, or direction of adjustment. The remaining non-parameterized commands are primarily toggles or simple activations (e.g., open Wi-Fi, change user password). More details can be found in the next section.

6.1.2 Task-Type and Parameter Distribution

We further analyze command structure by grouping samples along two axes: (1) task type (OS operation vs. question & answer) and (2) parameterization (with or without arguments). The summary statistics are shown in Table 2, from which we can observe that: (1) Parameterized tasks exhibit longer and more semantically complex descriptions, averaging 13.7 Chinese tokens, compared to 10.9 for non-parameterized ones; (2) Unique command coverage is higher for parameterized subsets (43 vs. 39), suggesting that parameterization is not confined to a few categories but broadly distributed across OS operations. This balanced yet diverse structure provides both lexical and functional heterogeneity—key properties for developing generalizable alignment models.

Table 2: Summary statistics by task type and parameterization in the ComAlign dataset. "Avg./Med. Length" denotes the average and medium Chinese token counts of the natural language descriptions

Type	Parameterized	Samples	Unique Commands	Avg./Med. Counts
OS Operation	False	838	39	10.9/10.0
OS Operation	True	868	43	13.7/13.0
QA Task	False	91	12	15.4/14.0
QA Task	True	14	7	17.8/17.0

6.2 Linguistic and Semantic Analysis

6.2.1 Lexical and Syntactic Variability

Chinese natural language descriptions exhibit extensive lexical diversity. Users employ multiple syntactic forms, ranging from imperative statements (e.g., "打开蓝牙", turn on Bluetooth), to declaratives with implicit intent ("我想开蓝牙", I'd like to open Bluetooth), to colloquial or incomplete forms ("蓝牙关一下", turn off Bluetooth). These variations lead to syntactic sparsity in the embedding space, i.e., surface forms

with the same intent often appear far apart in embedding space. For static vector models such as fastText, this results in reduced nearest-neighbor accuracy, as the embedding primarily reflects lexical co-occurrence rather than functional equivalence.

6.2.2 Semantic Ambiguity and Implicit Intent

Another challenge lies in implicit argumentation and vague modifiers. Chinese user inputs often express scalar changes through relative or metaphorical terms like “亮一点” (a bit brighter) or “声音调大一点” (make it a bit louder). Such expressions lack explicit numerical references, leaving the model to infer direction and magnitude purely from semantics. Parameter extraction becomes nontrivial in such cases. For example, while “调高亮度” (increase brightness) and “亮一点” (a bit brighter) are equivalent in effect, their lexical representations diverge significantly.

Similarly, semantically adjacent operations such as “设置屏幕超时时间” (set screen timeout) and “设置睡眠时间” (set sleep time) are nearly indistinguishable lexically but differ operationally. These examples underscore the semantic granularity problem, i.e., fine-grained OS operations often share overlapping vocabulary but distinct executable intents.

6.2.3 Length and Structural Trends

Fig. 7 depicts the distribution of descriptions’ Chinese token counts by parameterization. Parameterized descriptions are consistently longer and exhibit higher variance. This difference reflects the added cognitive and linguistic complexity of specifying optional parameters and contextual qualifiers. The median description length of parameterized commands is 13 Chinese tokens, compared to 10 for non-parameterized ones. This structural gap partially explains why baselines, especially VM and INVM, show lower alignment accuracy on parameterized subsets. The added contextual elements introduce both noise and ambiguity, which static vector embeddings are ill-equipped to capture.

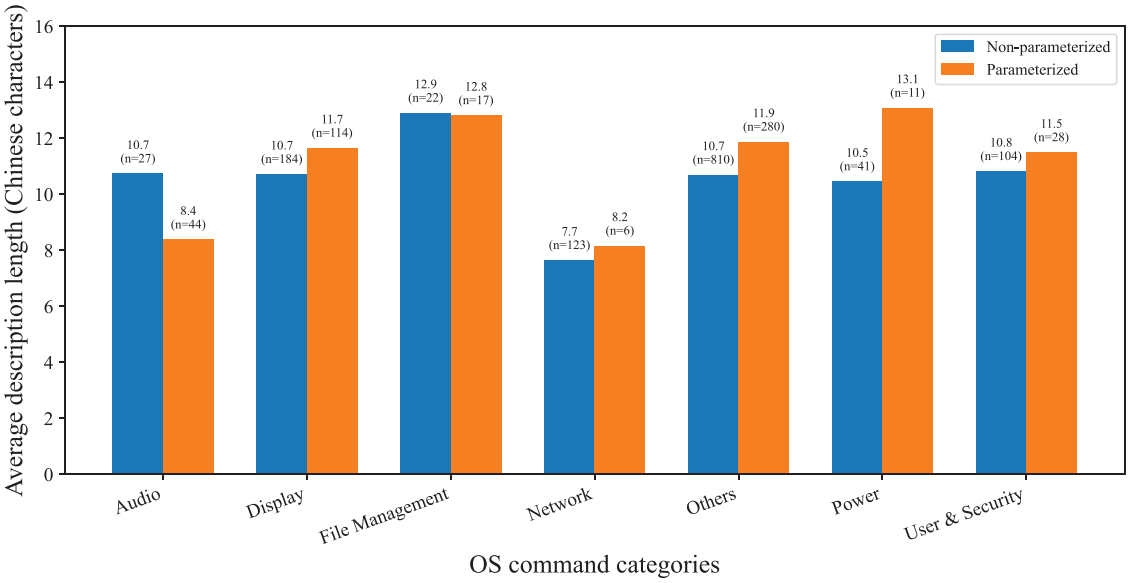


Figure 7: The average description length counted by Chinese tokens. We divide the natural language descriptions included in ComAlign into seven categories

6.2.4 Inter-Class Semantic Overlap

An additional layer of complexity arises from cross-command semantic interference. Commands belonging to the same functional family, such as display, sound, or network settings, tend to share highly overlapping terms like “设置” (setting), “调节” (adjust), “打开” (open), etc. The resulting inter-class proximity compresses the embedding space, causing misalignment between semantically adjacent but distinct commands. VMICE’s LLM-based intent-consistency evaluation alleviates this issue by reasoning over functional intent rather than relying solely on lexical similarity.

6.3 Error Analyses and Discussions

To gain deeper insight into failure modes, we conduct a qualitative analysis of 200 misaligned samples and present detailed error analysis below.

(1) **Paraphrase and synonym confusion.** A major source of error is lexical paraphrasing, i.e., different expressions referring to the same operation. For example, “开蓝牙” (turn on Bluetooth) and “启用蓝牙” (open Bluetooth) are semantically equivalent but lexically dissimilar. The VM baseline often fails due to insufficient embedding overlap, whereas the VMICE baseline handles most of these correctly.

(2) **Multi-Intent or compound commands.** Natural language descriptions like “调亮屏幕并打开护眼模式” (increase brightness and enable eye-protection mode) contain multiple actions. Since ComAlign currently associates one description with a single normalized command, such samples violate the one-to-one mapping assumption, leading to partial misalignment.

(3) **Implicit negation and polarity errors.** Natural language descriptions expressed with negation (e.g., “不要自动休眠”, don’t sleep automatically) or negated modality (“我不想关机”, I don’t want to shut down) are often misinterpreted, particularly when negation markers are subtle or colloquial (“别关了”, don’t shutdown; “先别休眠”, don’t sleep now). FastText-based embeddings fail to capture polarity, resulting in reversed matches.

(4) **Ambiguous parameter phrases.** Parameterized natural language descriptions often include vague adverbs like “稍微” (slightly) or “多一点” (a bit more). These expressions lack standardized quantitative mappings, making it difficult to judge whether “slightly brighter” corresponds to “+10%” or “+20%”. Current baselines ignore such fine-grained distinctions, treating all “increase” operations equally.

(5) **Semantic overlap across domains.** Natural language descriptions from semantically adjacent domains, such as display vs. power settings, often use overlapping lexemes (“亮度”, brightness; “休眠”, sleep; “屏幕”, screen). Without contextual grounding, the VM and INVM baselines often misclassify these cases. This observation highlights the potential benefit of introducing contextual embeddings (e.g., BERT-style models) or hierarchical intent taxonomies to disambiguate functional relations.

6.4 Limitations and Future Directions

Despite promising results, our study has several limitations that highlight both the complexity of aligning natural language with OS commands and the opportunities for future improvement.

(1) **Data scale and diversity.** ComAlign contains 1811 manually curated pairs across 82 OS command types, which remains limited in size compared to large English datasets such as NL2Bash or NLC2CMD. Some OS command categories are underrepresented, and the dataset primarily focuses on desktop-level operations. Future work will focus on validating the proposed approach

on larger-scale and more heterogeneous datasets, including data collected from real industrial or practical applications.

(2) **Modeling and representation constraints.** All three baselines rely on static word embeddings (fastText) to represent natural language descriptions, which are limited in capturing contextual semantics, compositional structure, and negation. Consequently, semantically similar expressions such as "调亮屏幕" (brighten the screen) and "把亮度提高" (increase brightness) may appear far apart in embedding space. Although the LLM-enhanced VMICE baseline improves performance (up to **46.9**% alignment accuracy), it introduces higher computational cost, dependency on prompt design, and limited reproducibility. Future work will explore adaptive parameter tuning strategies and alternative model architectures to further improve robustness and generalization performance.

Future work will investigate optimization techniques such as model compression, parallel computing, and incremental learning to reduce computational overhead.

(3) **Evaluation scope and practical deployment.** The current evaluation focuses solely on alignment accuracy between user inputs and normalized OS command descriptions, without assessing parameter extraction, compound intent handling, etc. Furthermore, real-world deployment of OS-level natural language agents introduces safety and policy challenges, such as access control, permission handling, and prevention of unintended operations. Future work will extend ComAlign to support end-to-end evaluation, including safety-aware and human-in-the-loop testing, to develop trustworthy and deployable intelligent OS systems.

In summary, ComAlign serves as an initial but essential step toward understanding and benchmarking natural language–to–command alignment. Overcoming these limitations will require expanding data coverage, enhancing semantic modeling, and integrating alignment evaluation into real-world system contexts.

7 Conclusion

In this paper, we present **ComAlign**, the first Chinese benchmark for aligning natural language descriptions with normalized operating system (OS) command descriptions as far as we know. The dataset contains 1811 annotated pairs across 82 types of OS commands, supporting research on aligning natural language with OS commands. We also propose three baselines, i.e., Vector Matching (VM), Input Normalisation and Vector Matching (INVM), and Vector Matching with Intent-Consistency Evaluation (VMICE). Experimental results show that the basic VM baseline achieves 39.2% accuracy, while the LLM-powered VMICE baseline raises performance to 46.9%. These results highlight both the promise and the difficulty of precise language–command alignment. Future work will expand ComAlign's data scope, explore stronger semantic encoders, and evaluate end-to-end models that jointly optimize accuracy, efficiency, and safety.

Acknowledgement: None.

Funding Statement: This work was supported by the National Key Research and Development Program under Grant 2024YFB4506200, the Science and Technology Innovation Program of Hunan Province under Grant 2024RC1048, and the National Key Laboratory Foundation Project under Grant 2024-KJWPDL-14.

Author Contributions: The first author Shasha Li conceived and supervised the study, and drafted the manuscript. The corresponding author Bin Ji performed the data collection, analysis, and interpretation. The corresponding author Jie Yu provided guidance, manuscript revision, and project management including funding acquisition. Xiaodong Liu and Jun Ma contributed to methodology design, validation of experimental results, and critical review and editing of the manuscript. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The ComAlign dataset is available from the corresponding author, Bin Ji, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. De Chaves SA, Benitti F. User-centred privacy and data protection: an overview of current research trends and challenges for the human-computer interaction field. ACM Comput Surv. 2025;57(7):1–36. doi:10.1145/3715903.
2. Sajja R, Sermet Y, Cikmaz M, Cwiertny D, Demir I. Artificial intelligence-enabled intelligent assistant for personalized and adaptive learning in higher education. Information. 2024;15(10):596. doi:10.3390/info15100596.
3. Wang L, Huang N, He Y, Liu D, Guo X, Sun Y, et al. Artificial Intelligence (AI) assistant in online shopping: a randomized field experiment on a livestream selling platform. Inf Syst Res. 2025;36(4):2358–74. doi:10.1287/isre.2023.0103.
4. Chen J, Liu Z, Huang X, Wu C, Liu Q, Jiang G, et al. When large language models meet personalization: perspectives of challenges and opportunities. World Wide Web. 2024;27(4):42. doi:10.1007/s11280-024-01276-1.
5. Raiaan MA, Mukta MS, Fatema K, Fahad NM, Sakib S, Mim MM, et al. A review on large language models: architectures, applications, taxonomies, open issues and challenges. IEEE Access. 2024;12:26839–74. doi:10.1109/access.2024.3365742.
6. Koubaa A, Ammar A, Boulila W. Next-generation human-robot interaction with ChatGPT and robot operating system. Softw Pract Exp. 2025;55(2):355–82. doi:10.1002/spe.3377.
7. Zhang C, Lu W, Ni C, Wang H, Wu J. Enhanced user interaction in operating systems through machine learning language models. In: Proceedings of the in International Conference on Image, Signal Processing, and Pattern Recognition (ISPP); 2024 Mar 8–10; Kunming, China. p. 1623–30.
8. Lin XV, Wang C, Zettlemoyer L, Ernst MD. NL2Bash: a corpus and semantic parser for natural language interface to the linux operating system. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018); 2018 May 7–12; Miyazaki, Japan.
9. Agarwal M, Chakraborti T, Fu Q, Gros D, Lin XV, Maene J, et al. Neurips 2020 nlc2cmd competition: translating natural language to bash commands. In: Proceedings of the NeurIPS 2020 Competition and Demonstration Track; 2020 Dec 6–12; Vancouver, BC, Canada. p. 302–24.
10. Fu Q, Teng Z, White J, Schmidt DC. A transformer-based approach for translating natural language to bash commands. In: Proceedings of the 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA); 2021 Dec 13–16; Pasadena, CA, USA. p. 1245–48.
11. Wang L, Zhang A, Wu K, Sun K, Li Z, Wu H, et al. DuSQL: a large-scale and pragmatic Chinese text-to-SQL dataset. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2020 Nov 16–20; Online. p. 6923–35.
12. Wang Y, Yang L, Chen B, Zou G, Xu K, Tang B, et al. Text2Mem: a unified memory operation language for memory operating system. arXiv:2509.11145. 2025.
13. Fu Q, Teng Z, Georgaklis M, White J, Schmidt DC. NL2CMD: an updated workflow for natural language to bash commands translation. arXiv:2302.07845. 2023.
14. Nair RP, Thushara MG. NL2Code: a hybrid NLP and model-driven framework for automated code generation from natural language and UML. In: Proceedings of the 2025 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS); 2025 Jan 18–19; Bhopal, India. p. 1–6.
15. Alharbi M, Alshayeb M. Automatic code generation techniques: a systematic literature review. Autom Softw Eng. 2026;33(1):4. doi:10.1007/s10515-025-00551-3.
16. Lu S, Guo D, Ren S, Huang J, Svyatkovskiy A, Blanco A, et al. CodeXGLUE: a machine learning benchmark dataset for code understanding and generation. In: Proceedings of the Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1); 2021 Feb 9; Virtual.

17. Feng Z, Guo D, Tang D, Duan N, Feng X, Gong M, et al. CodeBERT: a pre-trained model for programming and natural languages. In: Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020; 2020 Nov 16–20; Online Event. p. 1536–47.
18. Liu Z, Su J, Cai J, Yang J, Wu C. Instruct-code-llama: improving capabilities of language model in competition level code generation by online judge feedback. In: International Conference on Intelligent Computing. Singapore: Springer Nature Singapore; 2024. p. 127–37.
19. Kumar A, Sharma P. Openai codex: an inevitable future? Int J Res Appl Sci Eng Technol. 2023;11:539–43. doi:10.22214/ijraset.2023.49048.
20. Li Y, Choi D, Chung J, Kushman N, Schrittwieser J, Leblond R, et al. Competition-level code generation with alphacode. Science. 2022;378(6624):1092–7. doi:10.1126/science.abq1158.
21. Yang A, Yu B, Li C, Liu D, Huang F, Huang H, et al. Qwen2.5-1m technical report. arXiv:2501.15383. 2025.
22. Yang A, Li A, Yang B, Zhang B, Hui B, Zheng B, et al. Qwen3 technical report. arXiv:2505.09388. 2025.
23. Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, et al. Llama 2: open foundation and fine-tuned chat models. arXiv:2307.09288. 2023.
24. Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A, Letman A, et al. The llama 3 herd of models. arXiv:2407.21783. 2024.
25. Gallifant J, Fiske A, Levites Strekalova YA, Osorio-Valencia JS, Parke R, Mwavu R, et al. Peer review of GPT-4 technical report and systems card. PLoS Digit Health. 2024;3(1):e0000417. doi:10.1371/journal.pdig.0000417.
26. Hu S, Tu Y, Han X, He C, Cui G, Long X, et al. MiniCPM: unveiling the potential of small language models with scalable training strategies. arXiv:2404.06395. 2024.
27. Joulin A, Grave E, Bojanowski P, Mikolov T. Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics; 2017 Apr 3–7; Valencia, Spain. p. 427–31.