



MANUAL

• THE APPLICATION - AN HIGH-LEVEL PROJECT DESCRIPTION

The NONSENSE Generator is an innovative and versatile tool designed to analyze sentences and generate meaningful (or much more likely nonsensical) sentences based on user preferences. With a simple and intuitive interface, the application is accessible to users of all ages and expertise levels. Every feature is completely in English, ensuring universal usability and clarity in interactions.

The tool offers dynamic sentence manipulation through various strategies and patterns, allowing users to explore and generate sentence structures in unique and imaginative ways. Additionally, it integrates advanced mechanisms for linguistic processing such as word selection, sentence structure generation, and contextual adjustments based on predefined strategies (e.g., random structures, selected structures, tense-specific conjugations).

At its core, the application aims to provide an enjoyable and educational platform where creativity and logical sentence construction are tested and developed. Through additional features, such as detailed explanations and real-time modifications, users can better understand and experiment with sentence mechanics.

The main menu offers a user-friendly navigation system guiding users through the different functionalities such as sentence analysis, generation, and customization with tailored strategies.

The screenshot shows the NONSENSE Generator application window. The title bar reads "NONSENSE Generator". The menu bar includes "File", "Edit", and "Help". The main title "NONSENSE Generator" is centered. The interface is divided into two main sections. The left section, titled "Write your sentence to generate something", contains a text input field with the placeholder "This sentence is a good example", a blue "Analyze" button, and a checkbox labeled "Syntactic Tree". A "Use Generated" button is located at the top right of this section. The right section contains a "Generate" button, a "Sentence structure" section with radio buttons for "Random", "Same as analyze", and "Select" (with a dropdown menu showing "[noun] [v...]"). The "Word Generation" section has checkboxes for "All words new" and "Future Tense". Below these, there is a "Toxicity Levels" section with a checked checkbox and five input fields, each corresponding to a toxicity category: "Toxicity", "Profanity", "Insult", "Sexual", and "Politics". At the bottom left, there is a "Save" checkbox.



First Launch

The first time you launch our Generator, a Settings form will appear. As shown in the accompanying image, this form provides users with complete access to configure all input and output paths according to their preferences.

The user is required to specify the path to their personal API Key in the designated field within the Settings form.

It is mandatory to configure the API Key before using the application. Without specifying the API Key path in the Settings form, the application will not launch.

This API Key will later be utilized to establish a connection with the Google API, which the application uses for analyzing both input sentences and the sentences it generates.

Additionally, at the bottom of the Settings form, users will find various options to further personalize their experience. These options enable customization of key functionalities, ensuring the Generator aligns perfectly with individual preferences and workflow requirements.

Everything is now set up and ready to use. After clicking 'Apply,' a larger main form will appear. This form is divided into two distinct sections: **Analyze** and **Generate**.

First Launch

The application provides a variety of customizable settings that allow users to adapt its behavior to their needs. These include file paths for resources (e.g., API key, nouns, verbs, adjectives, sentence structures), numerical constraints like the maximum recursion level and sentence length, and the user interface theme (light or dark).

Additionally, you can customize paths for the log files, analyzed sentences, and generated sentences storage.

You can modify these settings anytime by opening the dedicated settings form through the appropriate menu. Changes made in the settings are applied to the application's functionality, ensuring a tailored user experience.

Analyze your sentence

Let's now focus on the left side of the screen: the **sentence analyzer**. This section is specifically designed to analyze the sentences provided by the user, offering tools to better understand the syntactic structure and details associated with each word

- At the top, you will find the *Input Text* section, where you can either type your own sentence or import a previously generated one from the generation section, which will be explained in detail later.
- The Analyze button performs a comprehensive syntactic analysis of your sentence. When clicked, it processes your text through natural language processing tools to identify and categorize each word according to its grammatical function (nouns, verbs, adjectives).
- The system extracts the sentence's structural components, generates a syntactic tree visualization showing how words relate to each other hierarchically, and displays the complete grammatical breakdown.
- Additionally, by enabling the Syntactic Tree option, you can visualize a hierarchical tree that breaks down each word and its associated syntactic meaning, showing how words relate to each other in the sentence structure.



The screenshot shows a web interface for sentence analysis. At the top, there's a text input field containing "This sentence is a good example" and a "Use Generated" button. Below the input is a blue "Analyze" button and a checked checkbox labeled "Syntactic Tree". The main area displays the analyzed sentence with grammatical tags: "This [noun] [verb] a [adjective] [noun]". Below this, a hierarchical syntactic tree is shown, starting with "Sentence" at the root, branching into "Noun Phrase" (containing "Determiner" for "This" and "Noun, singular" for "sentence") and "Verb Phrase" (containing "Verb, 3rd person singular present" for "is" and "Noun Phrase" for "a good example"). The "Noun Phrase" under "is" further branches into "Determiner" for "a", "Adjective" for "good", and "Noun, singular" for "example". At the bottom left, there is a "Save" button.

This kind of analysis helps you understand the linguistic architecture of your sentence, allowing you to see how different elements connect and function together.

Generate new sentence

The **generation section**, located on the right side of the form, allows users to customize various aspects of sentence creation. It provides a user-friendly interface where you can select specific options to tailor the output according to your preferences. From sentence structure to word selection, you have full control over the generation process. Below is a detailed list of the customization options available:

- **Sentence Structure Selection**

You can choose the strategy for structuring the generated sentences:

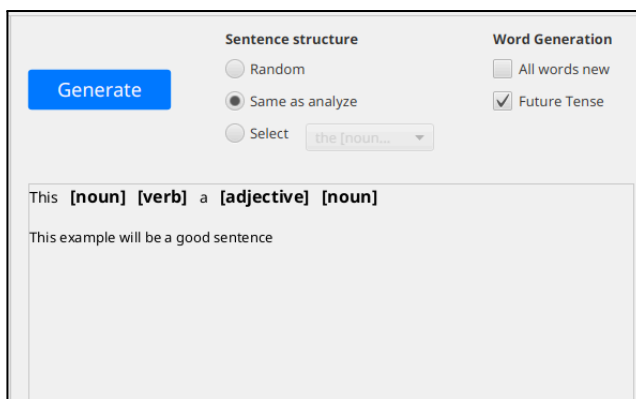
- *Random*: Generates a sentence using a randomly selected structure.
- *Same as analyze*: Maintains the sentence structure from the analyzed input.
- *Select*: Lets you select a specific sentence structure to use for generation.

- **Word Selection**

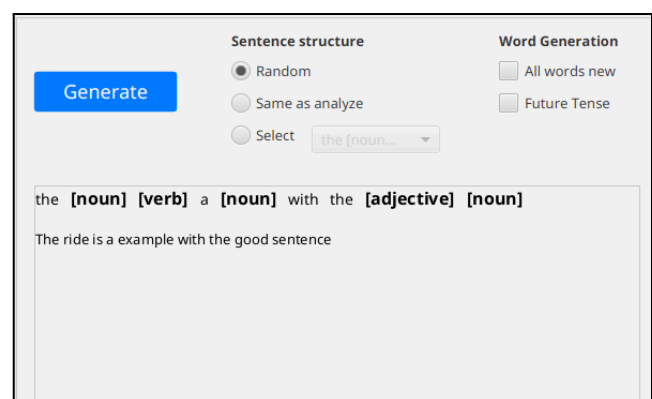
Customize how words are chosen for the generated sentence:

- *All New Words*: If unchecked the words of the analyzed sentence will be used, if checked every word is newly generated!
- *Future Tense Option*: verbs in the generated sentences will be automatically conjugated into the future tense, adding an additional layer of customization.

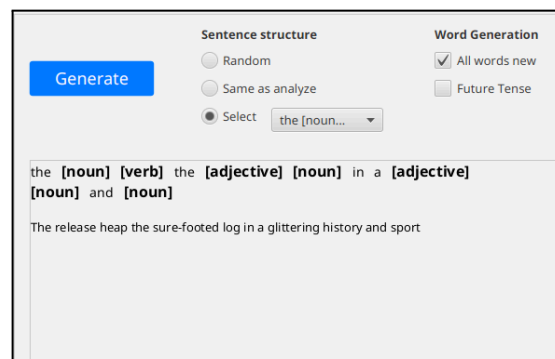
Below are some examples of generated sentences



The screenshot shows the 'Generate' button and the 'Sentence structure' section with 'Same as analyze' selected. The 'Word Generation' section has 'All words new' unchecked and 'Future Tense' checked. The generated sentence is: 'This [noun] [verb] a [adjective] [noun]'. Below it, a placeholder text reads: 'This example will be a good sentence'.



The screenshot shows the 'Generate' button and the 'Sentence structure' section with 'Random' selected. The 'Word Generation' section has 'All words new' unchecked and 'Future Tense' unchecked. The generated sentence is: 'the [noun] [verb] a [noun] with the [adjective] [noun]'. Below it, a placeholder text reads: 'The ride is a example with the good sentence'.



The screenshot shows the 'Generate' button and the 'Sentence structure' section with 'Select' selected. The 'Word Generation' section has 'All words new' checked and 'Future Tense' unchecked. The generated sentence is: 'the [noun] [verb] the [adjective] [noun] in a [adjective] [noun] and [noun]'. Below it, a placeholder text reads: 'The release heap the sure-footed log in a glittering history and sport'.



Moderation of the generated sentence

The generated sentences can be moderated across various categories, including toxicity, profanity, insults, sexual content, and political references. Activating the "Toxicity Levels" option allows the toxicity bars to visually represent these evaluation metrics. When enabled, the bars dynamically update to display the analyzed levels for each category. If disabled, the bars remain inactive and do not show or process any data.

Adding words - vocabulary

In this section of the application, users gain direct access to the vocabulary, which (as you already know) is organized into three categories: nouns, verbs, and adjectives.

To add new words, users simply need to input the desired words into the appropriate field (verbs, nouns, or adjectives). Multiple words can be added at once by separating them with spaces or commas. Once all words have been entered, a single click on the "Add" button will submit the words to the vocabulary. This intuitive process makes it easy to expand and refine the application's word database.

Saving process

In the bottom-left corner of the application, there is a "Save" option. When this option is selected, both analyzed and generated sentences are safely stored for future reference. The application appends the sentence to the respective file, such as the "analyzed sentences" or "generated sentences" storage files.

For more detailed *informations* you can also check out our *"The All-Functions Manual"* on this link for github: [github/NONSENSE-generator](https://github.com/NONSENSE-generator)



EXECUTION ENVIRONMENT REQUIREMENTS

The project requires:

- **Java Development Kit (JDK) 21:** the source and target compatibility are both set to Java 21 as specified in the file: `pom.xml`

```
<properties>
  <maven.compiler.source>21</maven.compiler.source>
  <maven.compiler.target>21</maven.compiler.target>
</properties>
```

This ensures compatibility with features introduced in Java 21.

- **JavaFX (version 21.0.7)** is required for the development of the graphical user interface (GUI). The required JavaFX modules: `javafx-controls`, `javafx-fxml`
To run the project with JavaFX, ensure that the Java environment is configured to support JavaFX.

Minimum System Requirements

- RAM** A minimum of **2GB**, as large linguistic models may consume significant memory.
- Processor** Any modern processor supporting Java 21.
- Disk Space** Approximately 500MB for runtime libraries, dependencies, and application files.

Operating System Compatibility:

The project can run on systems supporting Java 21 and JavaFX, including: Windows, macOS and Linux

Environment Variables

The application is designed to handle configuration in a secure and automated manner using predefined files and environment variables. Specifically, it relies on the file to dynamically manage key configuration paths and settings. **.env**

Key Variables Managed by the Application:

- **CONFIG_FILE_PATH:** Specifies the path to the primary configuration file used to store and load application settings.
- **DEFAULT_CONFIG_FILE_PATH:** Indicates the fallback configuration file path.
- **LOG_LEVEL:** Defines the verbosity level of logs generated by the application.



PROJECT DEPENDENCIES

The project uses Apache Maven for dependency management and builds. Maven handles the downloading and organization of the required third-party libraries, ensuring they are integrated seamlessly into the build process. Below is a list of critical dependencies and a brief explanation of their role in the project:

- **JavaFX Modules:** Used for building the graphical interface with tools like `javafx-controls` and `javafx-fxml` to ensure a responsive and interactive UI.
- **Log4J (Apache Logging):** Provides robust logging capabilities to track application actions, debug behavior, and handle runtime errors efficiently.
- **Stanford CoreNLP:** Offers advanced linguistic analysis, such as parsing the syntactic structure of sentences and recognizing parts of speech (e.g., nouns, verbs). The StanfordCoreNLP pipeline is primarily used to generate the syntax tree.

```
CoreDocument document = pipeline.processToCoreDocument(text);
CoreSentence coreSentence = document.sentences().get(0);
Tree tree = coreSentence.constituencyParse();
```

- **Google Cloud APIs:** Facilitates integration with Google's Natural Language Processing services, supporting text analysis, toxicity evaluation, and content moderation. The project uses Google Cloud's Natural Language API for content classification and text moderation. Here an example of the API Request

```
List<Token> tokens = new APIClient<AnalyzeSyntaxResponse>()
    .setSentenceToAPI(text)
    .setAPIType(APIClient.RequestType.SYNTAX)
    .execute()
    .getTokensList();
```

- **Apache Commons (Lang3):** Contains utility functions for string manipulation, randomization, and validating complex input data.
- **Dotenv:** Manages environmental configurations securely through an external file, ensuring flexibility for sensitive variables like API keys and file paths. `.env`
- **JUnit 5 (JUnit Jupiter):** Enables unit and integration testing, ensuring the application's codebase remains robust, reliable, and maintainable.

INTEGRATION OF EXTERNAL APIs

The project heavily integrates external APIs to perform linguistic analysis and content moderation efficiently. To streamline API interactions and ensure reusability, a dedicated utility class, **APIClient**, has been designed. This class manages all communication with external APIs, acting as a centralized component for handling service requests, configurations, and responses.

1. **Centralized API Handling:** The **APIClient** acts as a unified gateway for interacting with Google's Natural Language API. It abstracts the setup and execution of API requests, ensuring consistent usage across the project.
2. **Dynamic Configurability:** The class observes configuration changes (like API key updates) dynamically and adjusts the client behavior without requiring a restart.
3. **Singleton Instance Management:** The class ensures that the underlying `LanguageServiceClient`, which interacts with the Google API, is instantiated only once using the Singleton design pattern. This prevents redundant resource usage and ensures efficient communication.
4. **Supports Multiple Request Types:** The **APIClient** encapsulates different types of requests (MODERATION, SYNTAX) through an internal enum, delegating the appropriate logic for each use case.

INSTALLATION GUIDE

This guide describes how to run the *NONSENSE Generator* application either by downloading the pre-built JAR file or running the project from the source code using Maven.

1. Prerequisites

Before running the application, make sure you have the following tools installed and configured:

- **Java Development Kit (JDK) 21 or later**

Check the version with:

```
java -version
```

If not installed, download it from [Adoptium](#) or another trusted provider.

- **Apache Maven (Minimum version: 3.8.0)**

Verify the Maven installation with:

```
mvn -version
```

If Maven is not installed, download it at [Maven Official Website](#).

- **JavaFX Runtime (Version 21.0.7 or later)**

Download the JavaFX SDK from [OpenJFX Download Page](#), and extract it to a directory (e.g., `/opt/javafx-sdk-21.0.7`).



2. Running the Application

Option 1: Using the Pre-Built JAR

1. Download the JAR File

Navigate to the [GitHub Releases](#) page and download the latest pre-built JAR file (e.g., `NONSENSE-generator-1.0-SNAPSHOT.jar`).

2. Run the Application

Use the following command to launch the application, making sure to specify the **JavaFX SDK path**. For example if your JavaFX SDK is into `/opt/javafx-sdk-21.0.7`, the command would look like:

```
java --module-path /opt/javafx-sdk-21.0.7/lib \
  --add-modules javafx.controls,javafx.fxml \
  -jar NONSENSE-generator-1.0-SNAPSHOT.jar
```

Option 2: Running from Source with Maven

If you'd like to run the project directly from the source code, follow these steps:

1. Clone the Repository

Open a terminal and execute:

```
git clone https://github.com/your-repo-link.git
cd nonsense-generator
```

2. Run the Application via Maven

Use the following Maven command to launch the application:

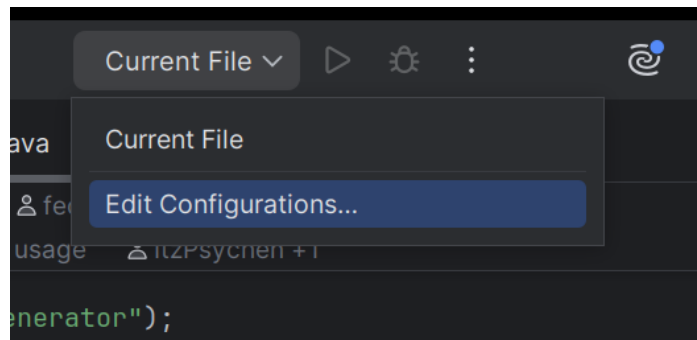
```
mvn javafx:run
```

Maven will automatically download dependencies and start the application.

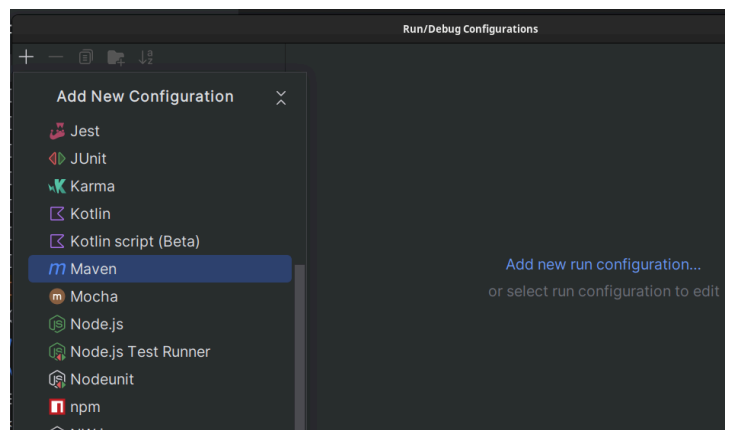
Option 3: Running from IDE (e.g. IntelliJ)

You can easily run the *NONSENSE Generator* application directly from your IDE, such as **IntelliJ IDEA**. Follow these steps to set up and execute the project:

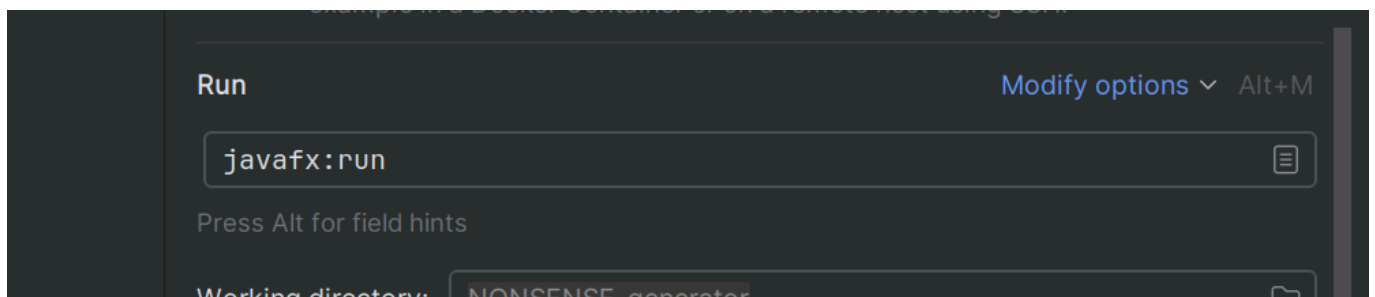
1. Open Edit Configurations...



2. Add New Configuration > maven



3. In the run option write javafx:run



If you encounter any problems or issues, refer to the project README or open a ticket in the GitHub Issues section.

That's all! You're ready to use the *NONSENSE Generator*.