OBSIDIAN: AN INTERACTIVE CHATBOT FOR CONTEXT-AWARE CONVERSATIONS

SYED SOFIYAN
Dept. of Cyber Security, Sreenidhi Institute of Science and Technology
21311A6204@sreenidhi.edu.in

KONGARAPU VAMSHI
Dept. of Cyber Security, Sreenidhi Institute of Science and Technology
21311A6222@sreenidhi.edu.in

PILLANAGOVILLA ARVIND
Dept. of Cyber Security, Sreenidhi Institute of Science and Technology
21311A6239@sreenidhi.edu.in

Mr. PEDDINTI MALLIKARJUN
Assistant Professor, Dept. of Cyber Security
mallikarjun.p@sreenidhi.edu.in

Abstract
Obsidian is a simple and light chatbot made using Python, Flask and NLP libraries. It is designed to give better chats by remembering what users say and replying smartly. Unlike old boring bots that forget everything after one message, Obsidian keeps the chat flowing. It is good for students, office work, and even casual chats. Obsidian uses basic memory tricks, easy query matching, and a friendly interface using Tkinter. The aim is to build a chatbot that is easy to make, use, and improve, without needing big computers or very hard coding.

Keywords: Chatbot, Obsidian, Python, NLP, Flask, Interactive UI, Context-Aware, Easy Chatbot

1.0 Introduction

Nowadays, as more and more people talk and work online, chatbots have became really important to help people talk to machines easily. Since so many users and websites are coming up everyday, it’s super important to have a system that can answer repeated questions and help users without needing a human all the time. One of the easiest ways to do this is by making a chatbot.

To fix the problem of old boring chatbots that don't really understand what people mean, we made a chatbot called Obsidian. It's small, light and smart, and is made using Python. This bot uses Natural Language Processing (NLP) and has a simple design with buttons and textboxes, so it can reply properly based on the topic and keep the chat going nicely.

The chatbot lets users chat with it live and it also remembers what was talked about before, so it doesn't sound weird after every new message. This makes talking to the bot feel much more natural and fun compared to normal bots that forget everything fast.

Obsidian also has a simple memory and tries to understand what the user wants to say. It can reply to hellos, common questions, and even learn new questions if users add them into a small text file. This makes it easy to use for schools, offices, and customer support.

Obsidian is made using Python and it uses Tkinter for the design part and Flask if you want to run it on the web. It also uses Python's own tools to save chat history, take inputs and reply back. The main idea of this project is to make something better than basic bots but not as hard as building super advanced AI. It's simple to make and fun to use!

2.0 Literature Review

Before we started working on our chatbot, we looked up many things about how chatbots are made and what problems they have. Many smart people already worked on this before. One of them is Géron. He talked about using machine learning tools like TensorFlow and Keras to make smart chatbots. These tools are very strong but also very heavy. They need big computers with lot of memory and fast processors, which we didn't have. Also, learning these tools fully takes lot of time and is very hard. So we decided to go with something more simple and easy to understand.Another very big change in chatbot world happened when Vaswani and his team made the Transformer model. This model was like a big invention because it helped machines to understand human language better. Today, almost every big AI chatbot like ChatGPT and Google's Meena uses Transformer models. But again, the problem with Transformer is that it needs very strong computers, a lot of memory, and many days of training. Also coding it is very very hard for small students like us. So instead of copying the whole thing, we picked up only some easy ideas like making the bot remember what user said earlier, and trying to give better replies that are connected to past conversation.We also learned from Jurafsky and Martin. They teached us some basics which are very important. Like how to break a sentence into small pieces (this is called tokenization), how to guess what a sentence really means (semantic parsing), and how to keep a conversation going properly. These basic ideas helped us a lot when we were designing Obsidian. We wanted our bot to not sound like a broken robot but sound a little more natural when chatting with users.

When we looked at real-world chatbots like Microsoft's XiaoIce and Google's Meena, we saw that they were very good in remembering what users said before. They even try to give replies which feel personal, like talking to a real friend. But we also found out that making chatbots like that is not easy at all. It needs very advanced programming skills,

lots of money, and access to very powerful computers. Also training those bots can take many weeks! So we thought that for Obsidian, we should keep it small and simple. We added a simple memory trick that lets the bot remember what the user says during one session. So until the chat is closed, the bot can understand and continue the talk properly. It is not super smart like Meena or Alexa, but for a school project, college work, and even small offices, it works pretty good.Another thing we saw is that many chatbots without memory feel very boring. If after every message the bot forgets everything, users feel frustrated. They don't want to repeat everything again and again. So making a bot that can at least remember some stuff during the chat was very important for us.We also learned that some big chatbots even understand user emotions and change the way they reply. For example, if a user is sad, the bot tries to cheer them up. We wanted to do something like that, but we realized that emotion detection is very hard and needs a lot of training data. So we didn't add that part now, maybe in future if we upgrade Obsidian.Besides that, we learned that user customization is very important too. Big chatbots allow users to add their own preferences, like topics or favorite things. So, we made Obsidian flexible, where users can add new patterns or questions into a simple text file. This makes it easy even for normal people who don't know coding to update the chatbot.

In the end, after reading and studying so many researches and trying small experiments, we learned one big thing. While it's very nice to know about big AI projects and technologies, it's much better to build something that fits what you can actually do. There is no point trying to make a super complex chatbot when you don't have big servers and a team of many people.So our Obsidian chatbot picks up the best ideas like memory, simple understanding of language, and easy custom changes — but keeps it light and simple. Anyone can use it easily without needing to study machine learning for years. We believe it's a good middle point between boring old bots and very high-level AI assistants.Maybe in the future, we can improve Obsidian even more by adding voice replies, better memory, or even connecting it to cloud servers. But for now, it works nice as a friendly chatbot that helps in school, colleges, and small businesses.

3.0 Statement of the Problem

Now a days, chat bots are use in many places like apps, websites and all. But most of them not work properly. Some bots are too simple, they don't remember anything after one message. So user feel like bot is dum and chat gets broken. Other bots are too much hard, they need big coding and super computers to run. Normal people cant make or use them easy.

Also, many chatbots are not easy to change. If we want to add new question or answer, we have to go into code and that is not easy. In schools or small office, people dont know coding much, so they can't do that. So it become big problem.

So we need a bot that is easy, smart and simple. It should remember user's chat and give good reply. That's why we make Obsidian. It is small chatbot but smart. It can remember what user told and reply nicely. It works good in normal computer also. People can add new questions from text file only, no need coding. So it is good for school, small work places and other use. Obsidian is for everyone who want chatbot but dont want hard work.

4.0 Research Objectives

To build a chatbot that remembers chats and gives better replies
Most chatbots forget what user said after one message. Our goal is to make a chatbot that can remember what user is saying during the chat, so that it can reply in a better and more smart way. This make the talk more natural and not broken like old bots.

To make it simple and lightweight using Python and Flask
Many chatbots need very big setup or heavy tools. But we want our chatbot to be easy to make and use. So we use Python, which is simple language, and Flask for web part. This way our chatbot works even on normal computers.

To allow easy customization for adding new questions and answers
We want normal users also to change or add things in chatbot. So we design it in a way where people can just add new questions and answers using a text file. No need to go into code. This helps schools and small teams who don't know programming.

5.0 Research Methodology

For making this Obsidian chatbot, we not follow very big or pro steps. We just plan simple way, so we can make it without getting stuck. First we think which all tools we can use. Python we already learnt a bit, so we pick that. It also have many library that help to make chatbot and all. So we no need search too much.
Then for front side, like the screen where we type and chat, we used Tkinter. It help to make a basic window where user can write and bot reply come. Also we added Flask, but that more for if we want to use it in website later. For now we not used Flask much. After that we use some NLP tools also, like NLTK and spaCy. These help in breaking user sentence and checking main words. But we not understand everything from those libraries, only basic use.
We divide our chatbot into 3 parts. First one is that NLP part, which check what user is trying to say. Second is memory part, where it keep some things from chat so bot remember little. Third is the window part where user type. This way, our bot don't act dumb after each message.
We made one Python dictionary. Inside it, we put questions like "what is your name" or "how are you" and give ready answers. So if user ask that, bot reply fast. Then we also make one file details.txt. Here user can add more questions and answers. Even people who don't know coding can open that file and write new lines. So anyone can add, not just us.
Then we write code for working of chatbot. When user type something, bot first check if it's "hello" or like that. If yes, then bot say hello too. Then it check if that message is in dictionary or text file. If it is, then bot say the correct reply. If not found, then it say sorry. If user say bye or something similar, bot end the chat and say goodbye.
We tested all this after we make the code. We typed many things like real talk, some wrong spellings also. We tried questions that are there, and some that not there.

When question is known, bot reply correctly. When unknown, it say sorry. When we say bye, it stop chat. So we feel okay it working.
We check the memory also. We type “my name is Ankit” then after some time we say “what’s my name” then bot say Ankit. This means it remembering things from same chat. But after chat end, it forget. That’s fine because we only put short memory in it.
We give this bot to 2-3 friends also. We tell them to try and add one new question in text file. They did, and bot start replying that also. So without coding, people can use it. That was our plan also. Simple and editable.
Then we check if it work on normal laptop. Our laptop has 4GB RAM and not high-end. Still it run fine, no hanging. So we think, okay it working even in simple system. We didn’t try in mobile, but maybe it work in future.
We also made small plan-like code, what they call pseudo code. It not real code but show how it work. From opening window, taking message, checking it, and giving reply or saying sorry. We write that also in our report.
We didn’t want to make some big AI bot. Just want something that don’t sound same every time and don’t forget everything after each message. Many chatbot are like that. So we focus on making it little smart with memory and easy for anyone to use.
So in short our steps was:

- Choose simple tools like Python and Tkinter
- Make chatbot in 3 parts: thinking, memory and typing part
- Put ready answers in file and allow user to edit it
- Check if chatbot talk properly
- Make sure it not need high system
- Give to others to test also

This full method help us to make Obsidian. It’s not advanced like big AI, but it’s good for schools, mini projects, or office chatbots. It reply nicely and don’t act stupid. In future we may try to add voice, or more smart brain, but for now, it working good and we happy with it.

6.0 Research Findings and Results

After we done building our chatbot, we tested many things to see if it's working proper or not. We chat with it like normal person and give different type of messages. Some messages bot already knows, and some were random to see what it do.

6.1 Chat with Matched Questions

When we give known question, bot answer nicely. Like if user say:

User: “What is your name?”
Bot: “My name is bot”

So it’s giving right reply. The chat window show both user and bot message clearly. It also keep showing the full chat like a scroll so nothing disappear. This help to see old message also when chat is long.

We also try greetings like “hello”, “hi”, and bot reply back with same kind of word. So this part was working good and we not find any mistake in it.

6.2 Chat with New or Unknown Questions
When user ask something that bot don't know, then it show default message. Like:

User: “Tell me a joke”
Bot: “Sorry”

This message come when bot not find answer in dictionary or file. It not try to give wrong or confusing reply. It just say sorry and move on. This way chat not become weird or wrong.

Even when we give spelling mistake or random text, chatbot not crash or hang. It just say sorry or nothing. So system is handling it smoothly and not breaking. This show that bot is safe and not acting strange when input is unknown.

6.3 Chat Memory Testing
We also check if bot remember what we saying. So we typed:

User: “My name is Rahul”
Then later we typed:
User: “What’s my name?”
Bot: “Rahul”

So memory is working in that time. But after we close chat and open again, then it forget. Which is okay because we made memory only for one session, not forever.

This help to make chat feel better and more like real talk, not like robot who forget everything in 1 second.

6.4 Interface Performance
The interface also working without lag. It open fast and not giving error. We tried pressing enter, using buttons, typing fast — still it working fine. On normal laptop also it run smooth. We didn’t use any powerful system or server, just tested in simple college laptop.

Scroll bar also work. When chat become long, we can go up and down to see old replies. This help in long chats. Input box is easy to use and reply come in proper format. There is no mix-up.

6.5 Editing and Updating
We also tested if someone else can add new questions. So our friends just opened that text file and added one line like:

What is Python? = Python is a programming language.

Then we typed that question in chat and bot replied with the new answer. So without touching code, chatbot started replying to new question. This show editing part is working and normal user also can do changes.

7.0 Conclusion

In this project we made a chatbot called Obsidian. Our main idea was to do something simple which can talk with user properly. We not try to make too big or advanced chatbot like Google one. This one is for school level or college small use only.

Many chatbot they forget what we say before. So we wanted to fix that. In our chatbot it remember some part of chat, like name and small things. This help for better talking, like if I say “my name is Raj” and later ask “what’s my name?” it give answer correctly. But after closing it forget, that is okay because we no use any big memory system.

We used only basic things for making this. Python language we used because easy to write. Tkinter we used for GUI, so we can type and see reply in chat window. Flask also added if we want to run in browser later. And for understanding message, we used NLTK and spaCy. We not use full features, only simple one we understood.

The bot is working good in normal laptop. No need of high RAM or graphic card. We test with many type of message like hello, how are you, and also some new question. If it know, it give answer. If not, it say sorry. It not crash or give wrong answer, so we are happy with that.

One more good thing is we can add new question-answer in text file. No need to open code. Anyone can do that even if they don’t know coding. This is useful for schools or office use where people want to update chatbot fast.

In future maybe we try to add voice system or more smart brain using AI model. But that is for later. Now this version is fine and doing what we want.

So final we say, Obsidian is small chatbot but very helpful. Easy to make, easy to use, and not boring like some old bots. We feel proud because we maked this with simple things and own effort.

8.0 Future Scope

Obsidian chatbot is doing okay, but lot of things can still be better. First, it can learn much more. If we add better machine learning, it will understand users more, and give them answers that are more personal. Also, if we get it to work on social media, WhatsApp, and smart devices, it will reach much more people, and be more helpful.

Another thing we can do is make Obsidian understand more languages. As the world is getting closer, it will be better if Obsidian talks in many languages, so more people can use it. Also, adding emotions to it will make it better. If Obsidian can know how person is feeling and talk like that, it will be more helpful, especially in customer service or mental health, where feelings matter a lot.

We also need to make sure Obsidian is fair and no bias is there. In future, we need to make sure it's giving answers to everyone without any wrong thinking. It should be like talking to a friend. Another thing we can do is make Obsidian better at making decisions. If it helps in solving problems or gives good advice, it can be used in daily life or at work. Lastly, if we let more people from all over help in improving Obsidian, it will grow faster. If developers help, it can get better in many ways and work in hospitals, schools, and even finance. In future, Obsidian can change how people talk with AI and help in many ways.

9.0 References

[1] Vaswanii, A., Shazir, N., Parmar, N., Uszkoret, J., Jons, L., Gomezz, A. A., Kaizer, Ł., & Polosukhinn, I. (2017). Attention is all you need. *In Advances in Neural Information Processing Systems (NeurIPS).*

[2] Bordes, A., & Wiston, J. (2016). Learning end to end goal-orientated dialouge. *In Proceeding of the 30th International Conference on Neural Information Processing Systems (NIPS 2016).*

[3] Chaturvadi, S., & Pal, P. (2018). Machine lerning based chatbots: A survey. *Internation Journal of Computer Applicatons.*

[4] Hancok, J., & O'Rourke, R. (2019). Chatbot techonlogy and aplication. *Internatiional Journal of Artifical Intelligence & Applciations (IJAIA).*

[5] Zhou, H., & Chiean, S. (2020). Review on Chattbot Technolgy: A Survey. *International Journal of Computer Applciations.*

[6] Shawar, B. A., & Atwell, E. (2007). Chatbots: Are they really usful? *In Proceeding of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies.*

[7] McTeer, M. (2017). The rise of converstional interface: A new type of user experiense. *Journal of Computin & Information Technoledge.*

[8] Radziwil, N., & Benton, M. C. (2017). Evaluating quailty of chatbots and intellignent assitants. *Computer Science & Information Systems.*

[9] Rashid, A., & Zainab, S. (2019). Sentiment analisys and emotion recogniton in chatbots: A survey. *Internatiional Journal of Computional Intelliegence and Applciations.*

[10] Mikolov, T., Chen, K., Corrado, G., & Den, J. (2013). Efficent estimation of word represantations in vector sapce. *In Proceeding of the International Conference on Learning Represntations (ICLR).*