CONSTRUCTION OF CHATBOTS AND ONTOLOGIES FOCUSED ON CORAL REEF FISH

Steve Desilets

MSDS 453: Natural Language Processing

Section 56 – Summer 2023

August 26, 2023

1. Introduction and Problem Statement

When scuba diving, divers may encounter many fish with which they are unfamiliar and which may have dangerous defense mechanisms, and divers may unknowingly present certain dangers to the integrity and viability of coral reef ecosystems. When divers have questions about the wildlife they witness underwater, they need a method to obtain answers to their questions. Fortunately, chatbots empowered by natural language processing pipelines enable users to easily find answers to their questions in highly cost-effective and efficient ways that don't require the manual attention of subject matter experts. While data scientists can construct chatbots to answer questions related to a broad array of domains, in this paper, we focus on creating chatbots to answer questions about coral reef fish. The corpus of 5,550 words leveraged to train the chatbots in this study is pulled directly from the "Coral reef fish" page on Wikipedia (Wikipedia 2023). After creating five different types of chatbots for this study, we assess the performance of each chatbot in providing accurate information, so that we understand which methods are most effective at answering questions about coral reef fish. Through these analyses, we hope to learn how to most effectively construct chatbots focused on empowering individuals – especially scuba divers – about coral reef fish ecosystems, so that humans can interact with them in a safe way for both themselves and for the wildlife.

2. Research Design and Modeling Methods

This study leveraged six methods of creating chatbots and knowledge graphs capable of enlightening users about coral reef fish.

After importing the text from Wikipedia's "Coral reef fish" webpage into a Jupyter Notebook, we created the first chatbot by leveraging a Sentence Transformer model that transformed each sentence in the corpus into a 384-dimensional embedding vector (Espejel 2022). This first chatbot then would similarly transform each question entered by users into similarly constructed semantic vector representations and calculate the cosine similarity between the question vector and each of the vector

representations of the corpus' sentences. Subsequently, the Sentence Transformer Model would return the corpus sentence associated with the highest cosine similarity score as its response to the chatbot user.

We then created a second chatbot by fine tuning the ChatGPT-2 model in Python using 50 sample training questions about coral reef fish whose answers could all be found within the corpus (OpenAI 2019). After finishing this fine-tuning process, this large language model was ready to encode and respond to each question entered by users.

The third chatbot, which we refer to as the TF-IDF Cosine Similarity Chatbot, begins by tokenizing the corpus by sentence. The chatbot then conducts data wrangling on each sentence in the corpus and in the user's question - including word tokenization, lemmatization, transformation to lowercase, and punctuation removal. The chatbot applies TF-IDF vectorization to each sentence, calculates the cosine similarity between each of the sentences, and returns the sentence from the corpus with the highest cosine similarity to the user's question (Kulkarni 2020).

The fourth chatbot that we created leveraged DistilBERT, a Transformer-Based language model (Sanh et al. 2019). The DistilBERT model has 40% fewer parameters than its predecessor, the BERT model, which allows the DistilBERT model to be much less expensive computationally while preserving 95% of BERT's level of performance. Like the BERT model, the DistilBERT model was pretrained to complete masked language modeling and next sentence prediction – enabling it to gain a level of natural language understanding that is useful for question answering purposes. After providing this DistilBERT model with our corpus from Wikipedia, the chatbot was ready to answer questions about coral reef fish.

The last chatbot that we created leveraged was a version of the Roberta Model that was developed for the purpose of answering questions (Chan et al. 2023). Like the previous model, the Roberta model is pretrained to complete masked language modeling, which enabled the Roberta Chatbot to gain a strong natural language understanding. However, the training times for Roberta Models are generally longer than the training times for comparable large language models, like DistilBERT. We provided the corpus of

text to the Roberta Model, so that it would then be able to answer questions about coral reef fish. After creating each of these five chatbots, we were able to test each chatbot's performance by asking each chatbot the same 20 questions whose answers were all available in the corpus of information about coral reef fish. We then manually scored each chatbot response as being either correct, partially correct, or incorrect for subsequent analysis of chatbot performance.

After creating these five chatbots, we then created a knowledge graph to represent many of the complex relationships that exist within coral reef ecosystems. This ontology, which we created with the Cypher code presented in Appendix D, visually represented 26 coral reef species, 6 defense mechanisms, 1 attacking mechanism, and 29 relationships between those nodes. Knowledge graphs like this can integrate well with NLP pipelines and can intuitively convey networks of relationships (like those that exist within coral reefs) to audiences. The output associated with the construction of this ontology is provided in Appendix C.

3. Results

The results from the chatbot and ontology construction analyses in this study are displayed in Appendices A – E. Appendix B displays the 20 questions used to test the chatbots, as well as the target correct answers derived directly from reading the corpus and the answers provided by each of the five chatbots. After manually assessing the accuracy of each chatbot response, we color coded each answer as green, yellow, or red to reflect whether the answers were correct, partially correct, or incorrect, respectively. Appendix A conveys performance metrics from this analysis in a stacked bar plot that clearly visualizes how well each of the five chatbots performed at answering our test questions. Appendix C presents an ontology that visualizes the complex network of relationships between coral reef fish species that are described in the corpus. To promote transparency and reproducibility of our results, Appendices D and E provide the Cypher and Python code leveraged to create the knowledge graph and to conduct the chatbot analyses as well.

4. Analysis and Interpretation

Analysis of the results from our chatbot and ontology development efforts, as displayed in the appendices, reveals many interesting findings. Perhaps the clearest summary of the chatbot model performances is displayed in Appendix A, which visualizes the percent of the 20 test questions answered correctly, partially correctly, and incorrectly by each of the five chatbots. A clear hierarchy of chatbot performances emerges from this chart. The chatbots ranked in order from strongest performance to worst performance on the test questions is: 1) the Roberta Chatbot, 2) the DistilBERT Chatbot, 3) the TF-IDF Cosine Similarity Chatbot, 4) the Sentence-Based Transformer Chatbot, and 5) the Fine-Tuned GPT2 Chatbot. Closer examination of the actual chatbot responses, as displayed in Appendix B, can reveal greater insights into the relative strengths and weaknesses of each of the five models, so we delve into those results below.

Jumping into the strongest performing chatbot's responses first, we see that the Roberta Chatbot benefitted from having a strong natural language understanding. This chatbot's elevated NLU enabled it to be the only chatbot to correctly answer that 6,000 – 8,000 species of fish live in coral reefs and that fish camouflage themselves to ambush prey. In fact, it's hard to find an example of a test question where this chatbot underperformed compared to the other models. The best example of an underperformance by this chatbot was when it responded to the question "Which species is blue" with "bluestripe snapper" (which are primarily yellow with stripes of blue), but even in this case, the Roberta Chatbot was still partially correct. Given that the Roberta Chatbot answered 80% of the test questions correctly or partially correctly, this chatbot was the clear winner of this study.

Answering 55% of the test questions correctly or partially correctly, the DistilBERT Chatbot performed the second most strongly in this study. Like the Roberta Chatbot, the DistilBERT Chatbot seemed to benefit from a solid natural language understanding. For example, while other chatbots stumbled on the question "What are common herbivorous fish" by answering with content related to how *common* sea anemones are or what an herbivore is, the DistilBERT and Roberta Model understood that

the prompt was asking the chatbot to name a specific species and it correctly responded with "yellow tang." The DistilBERT Chatbot's natural language understanding certainly was not perfect though. When asked how many species of fish live in coral reefs (for which the correct answer is 6,000 – 8,000), the DistilBERT Chatbot comically answered "four" – clearly misinterpreting the corpus' statement about the four largest groups of herbivorous reef fish being parrotfishes, damselfishes, rabbitfishes, and surgeonfishes. Despite some relatively small mistakes like this, the DistilBERT Model was able to answer many test questions correctly.

In a close third place finish was the TF-IDF Cosine Similarity Chatbot, which answered 50% of the test questions correctly or partially correctly – an impressive feat for a chatbot with no natural language understanding. In a limited number of cases, the TF-IDF model's focus on specific words enabled it to outperform the more advanced models. For example, when asked what species can sting people, the TF-IDF chatbot correctly mentioned the stinging abilities of sea anemones, while the Roberta Model only answered partially correctly with "stargazer" which has defense mechanisms that are semantically similar (but not identical) to stinging: electrocution and venom. This focus on TF-IDF metrics did lead this model astray in some other cases though – like when the model was asked which species have commensalistic relationships and instead of responding with specific species, the TF-IDF Cosine Similarity Chatbot returned a sentence that described three different types of relationships that coral reef species commonly have. Another finding worth pointing out is that the TF-IDF Chatbot could benefit from some entity co-reference resolution features – as evidenced when the chatbot was asked "What species engage in schooling" and it responded with a sentence beginning with "they" – where "they" referred to a schooling fish.

Finishing in fourth place, the Sentence-Based Transformer Chatbot exhibited a mediocre performance by answering 40% of the questions correctly or partially correctly. In one instance, this chatbot's focus on semantic similarity between sentences did allow it to return the best answer. Specifically, for the question about how fish get rid of parasites, this Sentence-Based Transformer

Chatbot responded with the most complete and correct answer. More often, though, this chatbot's focus on semantic embedding similarity between sentences caused it to underperform compared to other models. Frequently, this chatbot returned sentences related to the topic of interest that didn't actually answer the question – as evidenced by this chatbot's responses to questions about commensalistic and mutualistic relationships, schooling species, and why fish are colorful.

Finishing in last place, the Fine-Tuned GPT2 Chatbot answered only 35% of the questions correctly or partially correctly – and most of those answers were just partially correct. In some cases, this chatbot produced largely rambling and nonsensical answers, like when it responded to a question asking it to name a common herbivorous fish by repeating "Stokes' sea bass are carnivorous fish that eat fish that are not their own" three times in succession. For other questions, when the Fine-Tuned GPT2 Chatbot seemed to not know an answer, it seemed to try to repurpose an answer from the set of training questions – like when it repeatedly answered "blacktip reef sharks" or when it comically answered that fish use parasites to attract mates since blacktip reef sharks and mating partner attraction were mentioned in the training dataset. Still, this model did have one advantage over the other models – it seemed to be more capable of pulling in outside information than the other models did. This feature enabled the Fine-Tuned GPT2 Model to actually outperform the other models for the question about where coral reefs exist – correctly pointing out that they exist in tropical and subtropical environments – despite this not being stated in the corpus.

While these chatbots can respond to individual user questions about coral reef fish, the knowledge graph displayed in Appendix C performs well as visualizing the complex network of relationships that exist within coral reefs. Viewers of such an ontology can quickly make complex connections like "spinefoot rabbitfish can defend themselves with venom and schooling" and "striated frogfish are ambush predators that can inflate themselves for self-defense." While these relationships are easily visible in this knowledge graph, they might be less apparent to users of a chatbot. Though this ontology does not try to incorporate all ~7,000 coral reef fish species, it still demonstrates its value by

displaying relationships between a small sample of coral reef fish. In this way, ontologies, chatbots, and chatbots that leverage information in ontologies, could all represent great resources (with varying strengths and weaknesses) for educating individuals about coral reef fish.

5. Conclusions

The analyses conducted throughout this study help us conclude how to best empower individuals to learn about coral reefs via chatbots and knowledge graphs. Given that the Roberta and DistilBERT chatbots answered most of the test questions correctly or partially correctly, our results suggest that Question-Answering Chatbots like Roberta and DistilBERT represent the most promising chatbot construction methods of those examined in this study. Furthermore, the mediocre performance of the TF-IDF and Sentence Transformer Chatbots suggest that chatbot development methods that rely on calculating the cosine similarity between vector representations of questions and sentences in a corpus may have moderate potential for success. Meanwhile, the relatively poor performance of the Fine-Tuned GPT2 Model suggests that reliance on this chatbot development method may not be the best option for creating chatbots to answer questions about coral reef fish. Finally, we learned that the ability of knowledge graphs to represent complex networks of information means that ontologies could be awesome ways to store information about the dynamic, interconnected relationships that exist within coral reefs. In conclusion, these findings could be highly beneficial to data scientists interested in understanding how to construct the most optimal chatbot to empower users to learn about coral reef fish.

6. Directions for Future Work

While this study resulted in many useful insights that could help data scientists better understand how to optimally build chatbots focused on answering questions about coral reef fish, there certainly exist exciting opportunities for further on this subject.

One route for further research on this topic would be to tweak the parameters of these existing models to further optimize their performance. For example, data scientists could experiment with additional data wrangling and vectorization methods for these models, which could have major impact on

their performance. Another example of how data scientists could refine the existing models would be by modifying the input data. For example, data scientists could gather a larger corpus of information about coral reef fish or provide the fine-tuned GPT2 model with additional training questions, which could significantly improve the models' performance since many machine learning algorithms perform best when provided vast quantities of data. In fact, for this study, we initially pre-trained the fine-tuned GPT2 model with just 10 questions and the chatbot only answered 20% of the questions partially correctly. After increasing the number of training questions to 50 for this fine-tuned GPT2 model, the model answered 35% of the questions correctly or partially correctly. This 75% improvement in performance resulting from just adding 40 additional training questions emphasizes the significant impact that refining these models could have.

Another avenue for further research regarding how to build the best chatbot for questions about coral reef fish would be to experiment with building new types of models. For example, researchers could experiment with building models that leverage recurrent neural networks / long short-term memory models that would be capable of answering questions about fish (though these models might be limited to answering questions by selecting one-term answers from the corpus vocabulary unless modifications were developed) (Thorn 2019). Researchers could also leverage the power of state-of-the-art models by building models in Python that integrate with ChatGPT in a way that extracts answers from our corpus of information about coral reef fish while still taking advantage of the large language model's incredible natural language understanding (Hafiz 2023). A third option for creating new chatbot models to answer questions about coral reef fish would be to build a large language model capable of extracting answers, when appropriate, from our ontology of information about coral reef fish. All these methods for constructing new chatbot models, as well as the previously described methods for improving our existing chatbot models, represent exciting opportunities to help individuals – especially scuba divers – learn about coral reef fish so that humans and marine life can interact more safely and sustainably.

References

Chan, Branden, Timo Möller, Malte Pietsch, and Tanay Soni. 2023. "roberta-base for QA." *Hugging Face*. https://huggingface.co/deepset/roberta-base-squad2.

Espejel, Omar. 2022. "multi-qa-MiniLM-L6-cos-v1." *Hugging Face*. https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-cos-v1.

Hafiz, Irtiza. 2023. "Creating Your Own AI-Powered Second Brain: A Guide with Python and ChatGPT" Medium. Creating Your Own AI-Powered Second Brain: A Guide with Python and ChatGPT | by Irtiza Hafiz | Level Up Coding (gitconnected.com).

Kulkarni, Mandar. 2020. "NLP Chatbot using nltk." *Github*. https://github.com/mandar196/InstaBot/blob/master/NLP%20Chatbot%20usnig%20nltk.ipynb.

OpenAI. 2019. "Better language models and their implications." *Hugging Face*. https://openai.com/research/better-language-models.

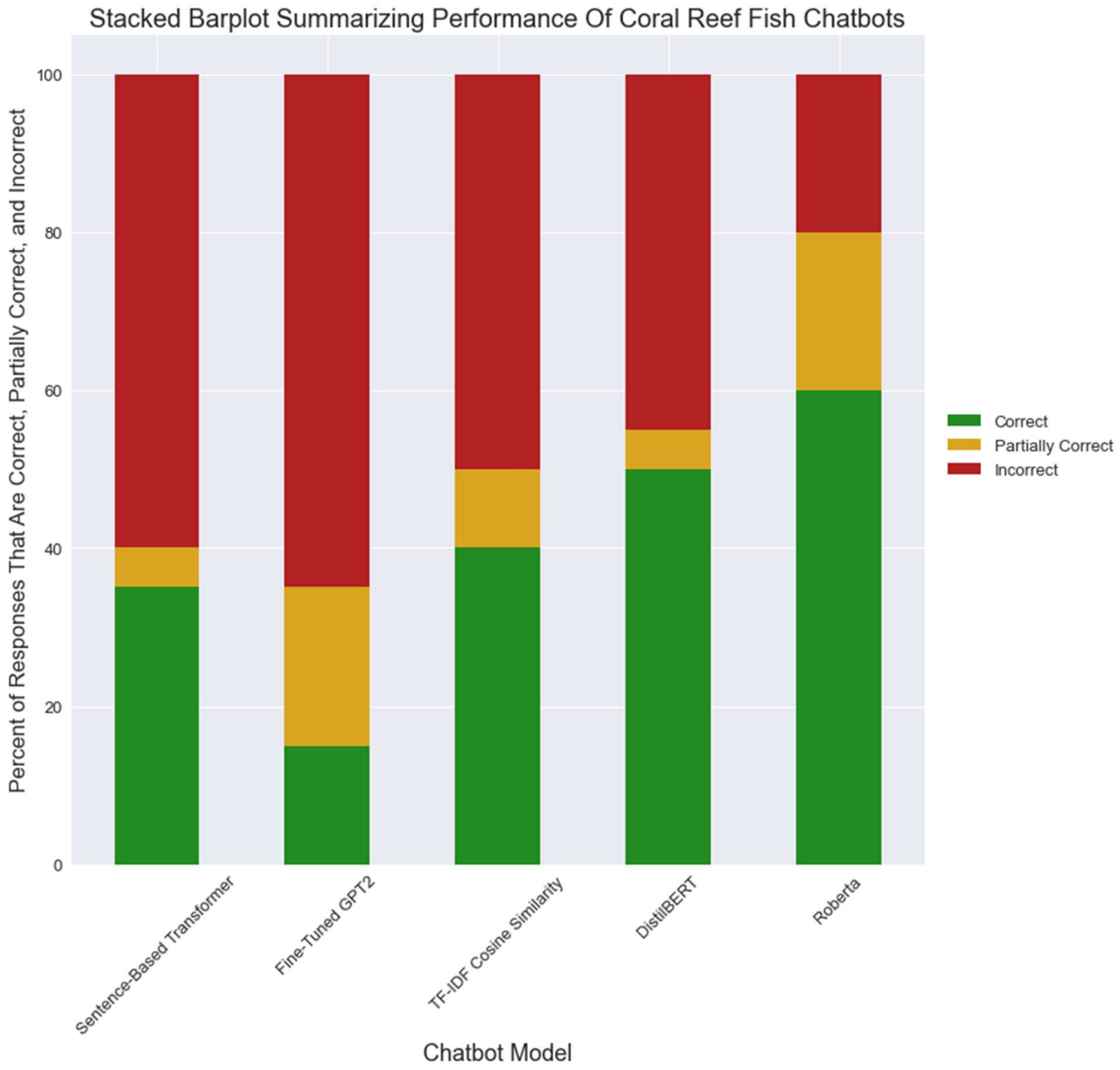
Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." *arXiv*. https://arxiv.org/abs/1910.01108.

Thorn, James. 2019. "Deep Learning for NLP: Creating a Chatbot with Keras." *Medium*. https://towardsdatascience.com/deep-learning-for-nlp-creating-a-chatbot-with-keras-da5ca051e051.

Wikipedia. 2023. "Coral Reef Fish" *Wikipedia*. https://en.wikipedia.org/wiki/Coral_reef_fish.

Appendix A – Chatbot Performance Metrics

The stacked bar plot below summarizes the performance metrics resulting from testing each of our chatbots with our 20 test questions. The percentage of questions answered correctly, partially correctly, and incorrectly are coded as green, yellow, and red, respectively in the chart.



Appendix B – Chatbot Answers To Test Questions

The table below displays the 20 test questions asked to each of the five chatbots, as well as their responses to each of those questions. For transparency, the target correct answer that we were looking for is provided as well. Cells are shaded as green, yellow, and red, to reflect whether the answers are correct, partially correct, or incorrect, respectively.

Questions	**Target Correct Answer**	**Sentence-Based Transformer Model Response**	**Fine-Tuned GPT2 Model Response**	**TF-IDF Cosine Similarity Model Response**	**DistilBERT Model Response**	**Roberta Model Response**
How many species of fish live in coral reefs?	6,000 - 8,000 species	the fish that inhabit coral reefs are numerous and diverse.	About 1,000 species of coral reef fish live in coral reefs.	coral reef fish are fish which live amongst or in close relation to coral reefs.	four	6,000â€“8,000
What is the most venomous fish?	Reef stonefish	the most venomous known fish is the reef stonefish.	blacktip reef shark	there is a distinction between poisonous fish and venomous fish.	reef stonefish	reef stonefish

What sharks live in coral reefs?	Whitetip, blacktip, and grey reef sharks (which are all species of requiem shark), Caribbean reef shark, as well as the Galapagos shark, the tawny nurse shark, and hammerheads	the whitetip reef shark almost exclusively inhabits coral reefs.	sharks. Sharks are the most abundant marine invertebrates on Earth. They are the largest marine invertebrates on Earth and are responsible for over half of all marine invertebrates. Sharks are also the most abundant marine invertebrates on Earth and are responsible for over half of all marine invertebrates. Sharks are also the most abundant marine invertebrates on Earth and are responsible for over half of all marine invertebrates.	coral reef fish are fish which live amongst or in close relation to coral reefs.	whitetip reef shark	caribbean reef shark
What fish are poisonous?	Giant moray and spotted trunkfish	there is a distinction between poisonous fish and venomous fish.	blacktip reef shark and blacktip reef shark	there is a distinction between poisonous fish and venomous fish.	coral reef fish	many reef fish
What fish can electrocute you?	Stargazer	some unmistakable contrasting patterns are used to warn predators that the fish has venomous spines or poisonous flesh.	blacktip reef shark	as with all fish, coral reef fish harbour parasites.	snapper	clown triggerfish

What species can sting you?	Fire Coral, Stingrays, Sea Anemones	the most venomous known fish is the reef stonefish.	sharks and rays	in return, the anemones provide the clownfish protection from their predators, who are not immune to anemone stings.	yellow stingrays	the stargazer
What species are parasitic?	nematodes, platyhelminthes (cestodes, digeneans, and monogeneans), leeches, parasitic crustaceans such as isopods and copepods, and various microorganisms such as myxosporidia and microsporidia)	monogenean parasites of the genus pseudorhabdosynochus (arrows) on the gill filament of a grouper.	a scorpionfish	parasites of coral reef fish include nematodes, platyhelminthes (cestodes, digeneans, and monogeneans), leeches, parasitic crustaceans such as isopods and copepods, and various microorganisms such as myxosporidia and microsporidia.	coral reef fish	damselfishes
What species are venomous?	Spinefoot rabbitfish, coral rabbitfish, reef stonefish, stargazer, lionfish	few of these venoms have been studied.	a scorpionfish	the most venomous known fish is the reef stonefish.	Spinefoot rabbitfish	the stargazer

What species are known for attacking scuba divers?	Lionfish can aggressively dart at scuba divers and attempt to puncture the facemask with their venomous spines	they are swift predators who patrol the reef in hunting packs.	sharks, rays, and other sharks.	lionfish can aggressively dart at scuba divers and attempt to puncture the facemask with their venomous spines.	lionfish	lionfish
What are common herbivorous fish?	The four largest groups of coral reef fishes that feed on plants are the parrotfishes, damselfishes, rabbitfishes, and surgeonfishes	herbivores feed on plants.	Stokes' sea bass. Stokes' sea bass are carnivorous fish that eat fish that are not their own. Stokes' sea bass are carnivorous fish that eat fish that are not their own. Stokes' sea bass are carnivorous fish that eat fish that are not their own.	sea anemones are common on reefs.	yellow tang	yellow tang
Why do fish camouflage themselves?	Ambush predation or Predator Avoidance	sometimes they camouflage the fish when the fish rests in places with the right background.	Because they are predators	sometimes they camouflage the fish when the fish rests in places with the right background.	when the fish rests in places with the right backgroun d	lets them ambush their prey
How do fish get rid of their parasites?	Some fish specialise as cleaner fish, and establish cleaning stations where other fish can come to have their parasites nibbled away	some fish specialise as cleaner fish, and establish cleaning stations where other fish can come to have their parasites nibbled away	They use parasites to attract mates	as with all fish, coral reef fish harbour parasites.	by biting, stinging, or stabbing	establish cleaning stations

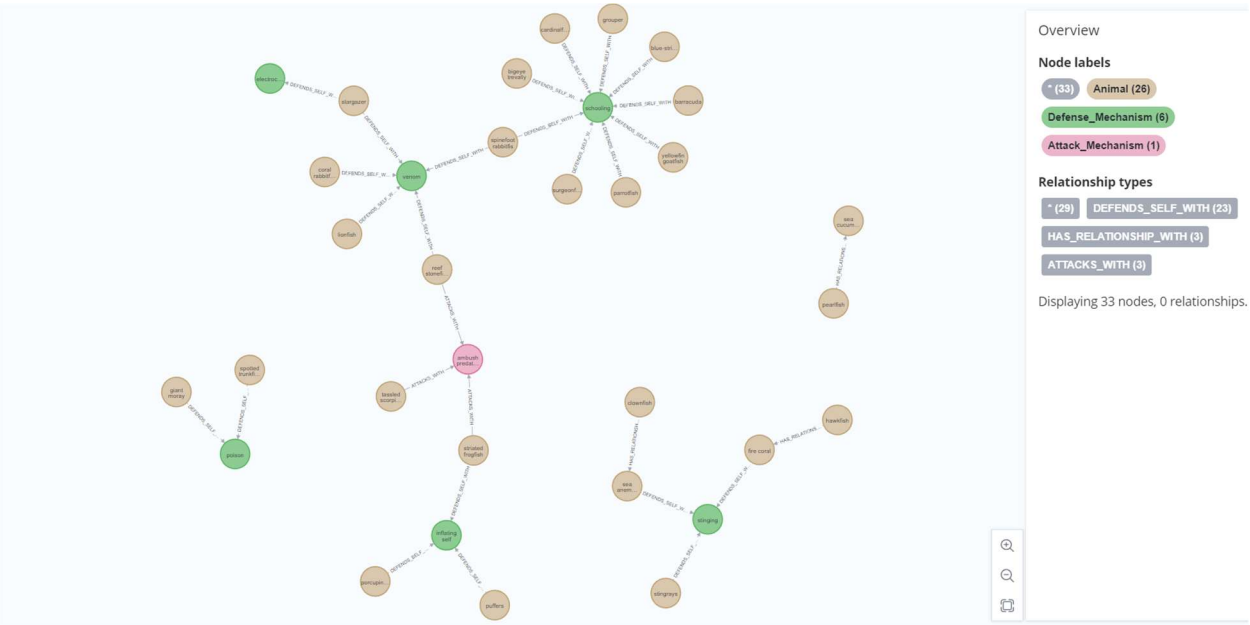
What species have mutualistic relationships?	Clownfish and Sea Anemones	the relationship can be mutualistic, when both species benefit from the relationship, commensalistic, when one species benefits and the other is unaffected, and parasitistic, when one species benefits, and the other is harmed.	sharks and sea urchins	there is a mutualistic relationship between sea anemones and clownfish.	sea anemones and clownfish	sea anemones and clownfish
What species have commensalistic relationships?	Hawkfish and Fire coral or Pearlfish and Sea Cucumber	the relationship can be mutualistic, when both species benefit from the relationship, commensalistic, when one species benefits and the other is unaffected, and parasitistic, when one species benefits, and the other is harmed.	giant butterfly and butterfly	the relationship can be mutualistic, when both species benefit from the relationship, commensalistic, when one species benefits and the other is unaffected, and parasitistic, when one species benefits, and the other is harmed.	sea anemones and clownfish	coral reef fish

Where in the world are coral reefs found?	The Indo-Pacific and the Tropical Western Atlantic	they are frequently found near the drop-offs at the outer edges of the reef, and less commonly within lagoons.	coral reefs are found in tropical and subtropical environments. They are the most abundant coral reef species in the world	coral reefs contain the most diverse fish assemblages to be found anywhere on earth, with perhaps as many as 6,000â€“8,000 species dwelling within coral reef ecosystems of the world's oceans.	top predators	indo-pacific
What species engage in schooling?	Parrotfish, Surgeonfish, Barracuda, Spinefoot Rabbitfish, Yellowfin Goatfish, Blue-Striped Snapper, Grouper, Cardinalfish, Bigeye Trevally	schooling fish have developed remarkable displays of precise choreography which confuse and evade predators.	sharks, which are the most common predator of sharks	they have evolved to find protection by schooling, sometimes with other species like shoaling rabbitfish.	spinefoot rabbitfish	the stargazer
Which species are ambush predators?	Reef Stonefish, Tasseled Scorpionfish, Striated Frogfish	another ambush predator is the striated frogfish (right).	a giant frogfish	the well camouflaged striated frogfish, a species of anglerfish, is an ambush predator.	frogfish	reef stonefish
Why are some coral reef fish colorful?	Recognition during mating, Warning Predators of Poison or Venom, Confusing Predators	coral reef fishes exhibit a huge variety of dazzling and sometimes bizarre colours and patterns.	Because they are not as bright as other reef fish.	coral reef fish are fish which live amongst or in close relation to coral reefs.	high biodiversity	coral reefs form complex ecosystems with tremendous biodiversity

Which species is blue?	synchiropus splendidus	the psychedelic synchiropus splendidus is one of only two animal species known to have blue colouring because of cellular pigment.	a blackfish	the psychedelic synchiropus splendidus is one of only two animal species known to have blue colouring because of cellular pigment.	bluestripe snapper	bluestripe snapper
What is the slowest species that lives in coral reefs?	sea horses	the slowest-moving fishes are the sea horses, often found in reefs.	sharks	some of these fish parasites have heteroxenous life cycles (i.e.	dwarf seahorse	dwarf seahorse

Appendix C – Ontology of Information About Coral Reef Fish

The knowledge graph below depicts information regarding coral reef fish that is conveyed within our corpus that we leveraged to create our chatbots.



Appendix D – Cypher Code Leveraged To Create The Knowledge Graph

Below, please find the Cypher code leveraged to create the ontology of information about coral reef fish.

```
// Create Animals

CREATE (a1:Animal {name:'clownfish'})
CREATE (a2:Animal {name:'reef stonefish'})
CREATE (a3:Animal {name:'stingrays'})
CREATE (a4:Animal {name:'spinefoot rabbitfish'})
CREATE (a5:Animal {name:'coral rabbitfish'})
CREATE (a6:Animal {name:'stargazer'})
CREATE (a7:Animal {name:'lionfish'})
CREATE (a8:Animal {name:'giant moray'})
CREATE (a9:Animal {name:'spotted trunkfish'})
CREATE (a10:Animal {name:'puffers'})
CREATE (a11:Animal {name:'striated frogfish'})
CREATE (a12:Animal {name:'porcupinefish'})
CREATE (a13:Animal {name:'fire coral'})
CREATE (a14:Animal {name:'sea anemone'})
CREATE (a15:Animal {name:'hawkfish'})
CREATE (a16:Animal {name:'pearlfish'})
CREATE (a17:Animal {name:'sea cucumber'})
CREATE (a18:Animal {name:'parrotfish'})
CREATE (a19:Animal {name:'surgeonfish'})
CREATE (a20:Animal {name:'barracuda'})
CREATE (a21:Animal {name:'yellowfin goatfish'})
CREATE (a22:Animal {name:'blue-striped snapper'})
CREATE (a23:Animal {name:'grouper'})
CREATE (a24:Animal {name:'cardinalfish'})
CREATE (a25:Animal {name:'bigeye trevally'})
CREATE (a26:Animal {name:'tassled scorpionfish'})

// Create Defense Mechanisms

CREATE (d1:Defense_Mechanism {name:'venom'})
CREATE (d2:Defense_Mechanism {name:'electrocution'})
CREATE (d3:Defense_Mechanism {name:'poison'})
CREATE (d4:Defense_Mechanism {name:'inflating self'})
```

```
CREATE (d5:Defense_Mechanism {name:'stinging'})
CREATE (d6:Defense_Mechanism {name:'schooling'})

// Create Attack Mechanisms

CREATE (k1:Attack_Mechanism {name:'ambush predation'})


// Create DEFENDS_SELF_WITH relationship
CREATE (a2)-[:DEFENDS_SELF_WITH]->(d1)
CREATE (a4)-[:DEFENDS_SELF_WITH]->(d1)
CREATE (a5)-[:DEFENDS_SELF_WITH]->(d1)
CREATE (a6)-[:DEFENDS_SELF_WITH]->(d1)
CREATE (a7)-[:DEFENDS_SELF_WITH]->(d1)
CREATE (a6)-[:DEFENDS_SELF_WITH]->(d2)
CREATE (a8)-[:DEFENDS_SELF_WITH]->(d3)
CREATE (a9)-[:DEFENDS_SELF_WITH]->(d3)
CREATE (a10)-[:DEFENDS_SELF_WITH]->(d4)
CREATE (a11)-[:DEFENDS_SELF_WITH]->(d4)
CREATE (a12)-[:DEFENDS_SELF_WITH]->(d4)
CREATE (a3)-[:DEFENDS_SELF_WITH]->(d5)
CREATE (a13)-[:DEFENDS_SELF_WITH]->(d5)
CREATE (a14)-[:DEFENDS_SELF_WITH]->(d5)
CREATE (a18)-[:DEFENDS_SELF_WITH]->(d6)
CREATE (a19)-[:DEFENDS_SELF_WITH]->(d6)
CREATE (a20)-[:DEFENDS_SELF_WITH]->(d6)
CREATE (a21)-[:DEFENDS_SELF_WITH]->(d6)
CREATE (a22)-[:DEFENDS_SELF_WITH]->(d6)
CREATE (a23)-[:DEFENDS_SELF_WITH]->(d6)
CREATE (a24)-[:DEFENDS_SELF_WITH]->(d6)
CREATE (a25)-[:DEFENDS_SELF_WITH]->(d6)
CREATE (a4)-[:DEFENDS_SELF_WITH]->(d6)

// Create Relationship Relationship
CREATE (a16)-[:HAS_RELATIONSHIP_WITH {type:'Commensalistic'}]->(a17)
CREATE (a15)-[:HAS_RELATIONSHIP_WITH {type:'Commensalistic'}]->(a13)
CREATE (a1)-[:HAS_RELATIONSHIP_WITH {type:'Mutualistic'}]->(a14)

// Create ATTACKS_WITH relationship
CREATE (a2)-[:ATTACKS_WITH]->(k1)
CREATE (a26)-[:ATTACKS_WITH]->(k1)
CREATE (a11)-[:ATTACKS_WITH]->(k1)

RETURN *
```

Appendix E - Python Code To Create Coral Reef Fish Chatbots

Steve Desilets

August 26, 2023

1) Chatbot 1 - Sentence-Based Transformer Chatbot That Leverages Cosine Similarity

In [41]:
```
import pandas as pd
import numpy as np
import re
import string
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from collections import Counter
from dataclasses import dataclass
from timeit import default_timer as timer
import random
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.tokenize import word_tokenize

import gensim
from gensim.models import Word2Vec

import spacy
from spacy import displacy

from spacy.matcher import Matcher
from spacy.tokens import Span

import networkx as nx
import matplotlib.pyplot as plt
from tqdm import tqdm
!pip install sentence-transformers
from sentence_transformers import SentenceTransformer, util
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity


from IPython.display import display, HTML

from typing import List, Callable, Dict, Tuple, Set

pd.set_option('max_colwidth', 600)
pd.set_option('display.max_rows', 500)
```

```
Requirement already satisfied: sentence-transformers in c:\users\steve\anaconda3\lib
\site-packages (2.2.2)
Requirement already satisfied: transformers<5.0.0,>=4.6.0 in c:\users\steve\anaconda3
\lib\site-packages (from sentence-transformers) (4.31.0)
Requirement already satisfied: tqdm in c:\users\steve\anaconda3\lib\site-packages (fr
om sentence-transformers) (4.66.1)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\steve\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\steve\anaconda3\lib\site-pa
ckages)
```

```
Requirement already satisfied: torch>=1.6.0 in c:\users\steve\anaconda3\lib\site-pack
ages (from sentence-transformers) (2.0.1)
Requirement already satisfied: torchvision in c:\users\steve\anaconda3\lib\site-packa
ges (from sentence-transformers) (0.15.2)
Requirement already satisfied: numpy in c:\users\steve\anaconda3\lib\site-packages (f
rom sentence-transformers) (1.22.4)
Requirement already satisfied: scikit-learn in c:\users\steve\anaconda3\lib\site-pack
ages (from sentence-transformers) (1.3.0)
Requirement already satisfied: scipy in c:\users\steve\anaconda3\lib\site-packages (f
rom sentence-transformers) (1.7.3)
Requirement already satisfied: nltk in c:\users\steve\anaconda3\lib\site-packages (fr
om sentence-transformers) (3.7)
Requirement already satisfied: sentencepiece in c:\users\steve\anaconda3\lib\site-pac
kages (from sentence-transformers) (0.1.99)
Requirement already satisfied: huggingface-hub>=0.4.0 in c:\users\steve\anaconda3\lib
\site-packages (from sentence-transformers) (0.16.4)
Requirement already satisfied: filelock in c:\users\steve\anaconda3\lib\site-packages
(from huggingface-hub>=0.4.0->sentence-transformers) (3.6.0)
Requirement already satisfied: fsspec in c:\users\steve\anaconda3\lib\site-packages
(from huggingface-hub>=0.4.0->sentence-transformers) (2023.6.0)
Requirement already satisfied: requests in c:\users\steve\anaconda3\lib\site-packages
(from huggingface-hub>=0.4.0->sentence-transformers) (2.31.0)
Requirement already satisfied: pyyaml>=5.1 in c:\users\steve\anaconda3\lib\site-packa
ges (from huggingface-hub>=0.4.0->sentence-transformers) (6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\steve\anaconda3
\lib\site-packages (from huggingface-hub>=0.4.0->sentence-transformers) (4.7.1)
Requirement already satisfied: packaging>=20.9 in c:\users\steve\anaconda3\lib\site-p
ackages (from huggingface-hub>=0.4.0->sentence-transformers) (21.3)
Requirement already satisfied: sympy in c:\users\steve\anaconda3\lib\site-packages (f
rom torch>=1.6.0->sentence-transformers) (1.10.1)
Requirement already satisfied: networkx in c:\users\steve\anaconda3\lib\site-packages
(from torch>=1.6.0->sentence-transformers) (2.7.1)
Requirement already satisfied: jinja2 in c:\users\steve\anaconda3\lib\site-packages
(from torch>=1.6.0->sentence-transformers) (2.11.3)
Requirement already satisfied: colorama in c:\users\steve\anaconda3\lib\site-packages
(from tqdm->sentence-transformers) (0.4.6)
Requirement already satisfied: regex!=2019.12.17 in c:\users\steve\anaconda3\lib\site
-packages (from transformers<5.0.0,>=4.6.0->sentence-transformers) (2022.3.15)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in c:\users\steve\an
aconda3\lib\site-packages (from transformers<5.0.0,>=4.6.0->sentence-transformers)
(0.13.3)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\steve\anaconda3\lib\sit
e-packages (from transformers<5.0.0,>=4.6.0->sentence-transformers) (0.3.2)
Requirement already satisfied: click in c:\users\steve\anaconda3\lib\site-packages (f
rom nltk->sentence-transformers) (8.0.4)
Requirement already satisfied: joblib in c:\users\steve\anaconda3\lib\site-packages
(from nltk->sentence-transformers) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\steve\anaconda3\lib\s
ite-packages (from scikit-learn->sentence-transformers) (2.2.0)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in c:\users\steve\anaconda3\lib
\site-packages (from torchvision->sentence-transformers) (9.0.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\steve\anaconda3\l
ib\site-packages (from packaging>=20.9->huggingface-hub>=0.4.0->sentence-transformer
s) (3.0.4)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\steve\anaconda3\lib\site-
packages (from jinja2->torch>=1.6.0->sentence-transformers) (2.0.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\steve\anaconda3\l
ib\site-packages (from requests->huggingface-hub>=0.4.0->sentence-transformers) (2.0.
4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\steve\anaconda3\lib\site-pack
```

```
ages (from requests->huggingface-hub>=0.4.0->sentence-transformers) (3.3)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\steve\anaconda3\lib\sit
e-packages (from requests->huggingface-hub>=0.4.0->sentence-transformers) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\steve\anaconda3\lib\sit
e-packages (from requests->huggingface-hub>=0.4.0->sentence-transformers) (2021.10.8)
Requirement already satisfied: mpmath>=0.19 in c:\users\steve\anaconda3\lib\site-pack
ages (from sympy->torch>=1.6.0->sentence-transformers) (1.2.1)
```

In [42]:
```
#Load Sentence Transformer model optimized for  sentence cosine similarity calculation

#The models below fully downloaded in Google Colab. This is the version of the google
#it was open in anoconda to be saved as pdf and the download graphics did not transfer
#it seems like it didn't download. However, it did in the orignal google colab noteboo
#analysis was run.

model = SentenceTransformer('multi-qa-MiniLM-L6-cos-v1')
```

In [3]:
```
#from google.colab import drive
#drive.mount('/content/gdrive')
```

In [43]:
```
# Only run this once, they will be downloaded.
nltk.download('stopwords',quiet=True)
nltk.download('wordnet',quiet=True)
nltk.download('punkt',quiet=True)
nltk.download('omw-1.4',quiet=True)
```

Out[43]:
```
True
```

In [44]:
```
#read in data
CORPUS_PATH = 'C:/Users/steve/OneDrive/Desktop/Github/Natural_Language_Processing/Chat
f=open(CORPUS_PATH,'r',errors = 'ignore')
raw=f.read()
raw=raw.lower()# converts to lowercase

#create list of sentences and words
sent_tokens = nltk.sent_tokenize(raw)# converts to list of sentences
word_tokens = nltk.word_tokenize(raw)# converts to list of words
```

In [45]:
```
#create greetings and greetings function

GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's up","hey",)
GREETING_RESPONSES = ["Hello"]


# Checking for greetings
def greeting(sentence):
    """If user's input is a greeting, return a greeting response"""
    for word in sentence.split():
        if word.lower() in GREETING_INPUTS:
            return random.choice(GREETING_RESPONSES)
```

In [46]:
```
# Generating response function
def response(user_response):
    chatbot_response=''
    sentence_encodings=model.encode(sent_tokens, convert_to_tensor=True)# generate ser
    sentence_encodings=sentence_encodings.cpu()
    vals = cosine_similarity(sentence_encodings[-1].reshape(1, -1), sentence_encodings
    #in the next cell adds the question as the last sentence of the sentence tokens, b
```

```
    #The code takes the last sentence (which is the question) and gets cosine similari
    #including itself
    idx=vals.argsort()[0][-2] #gets the index of the second highest similarity (the fi
    flat = vals.flatten()#reduces dimension of cosine similarity array to be able to s
    flat.sort() #sort the cosine similarity values
    second_cos_sim_val = flat[-2] #get the second highest cosine similarity value.
    if(second_cos_sim_val==0): #check the second highest cosine similarity value. If i
        #else return highest cosine similarity sentence.
        chatbot_response=chatbot_response+"Sorry, I do not have an answer to your ques
        return chatbot_response
    else:
        chatbot_response = chatbot_response+sent_tokens[idx] #use index of highest cos
        return chatbot_response
```

In [47]:

```
#Chatbot interaction code

flag=True
print("Welcome to the Informational Chatbot About Coral Reef Fish. To end session plea
print("\n")

while(flag==True):
    user_response = input()
    user_response=user_response.lower()
    if user_response!='exit':
        if(user_response=='thanks' or user_response=='thank you' ):
            flag=False
            print("Answer: You are welcome!")
        else:
            if(greeting(user_response)!=None):
                print("Answer: "+greeting(user_response))
            else:
                sent_tokens.append(user_response)
                word_tokens=word_tokens+nltk.word_tokenize(user_response)
                final_words=list(set(word_tokens))
                print("Answer: ",end="")
                print(response(user_response))
                print("\n")
                sent_tokens.remove(user_response)
    else:
        flag=False
        print("Thank you for using the Informational Chatbot About Coral Reef Fish. Go
```

```
Welcome to the Informational Chatbot About Coral Reef Fish. To end session please typ
e exit.


How many species of fish live in coral reefs?
Answer: the fish that inhabit coral reefs are numerous and diverse.


What is the most venomous fish?
Answer: the most venomous known fish is the reef stonefish.


What sharks live in coral reefs?
Answer: the whitetip reef shark almost exclusively inhabits coral reefs.


What fish are poisonous?
Answer: there is a distinction between poisonous fish and venomous fish.


What fish can electrocute you?
Answer: some unmistakable contrasting patterns are used to warn predators that the fi
sh has venomous spines or poisonous flesh.


What species can sting you?
Answer: the most venomous known fish is the reef stonefish.


What species are parasitic?
Answer: monogenean parasites of the genus pseudorhabdosynochus (arrows) on the gill f
ilament of a grouper.


What species are venomous?
Answer: few of these venoms have been studied.


What species are known for attacking scuba divers?
Answer: they are swift predators who patrol the reef in hunting packs.


What are common herbivorous fish?
Answer: herbivores feed on plants.


Why do fish camouflage themselves?
Answer: sometimes they camouflage the fish when the fish rests in places with the rig
ht background.


How do fish get rid of their parasites?
Answer: some fish specialise as cleaner fish, and establish cleaning stations where o
ther fish can come to have their parasites nibbled away.


What species have mutualistic relationships?
Answer: the relationship can be mutualistic, when both species benefit from the relat
ionship, commensalistic, when one species benefits and the other is unaffected, and p
arasitistic, when one species benefits, and the other is harmed.
```

```
What species have commensalistic relationships?
Answer: the relationship can be mutualistic, when both species benefit from the relat
ionship, commensalistic, when one species benefits and the other is unaffected, and p
arasitistic, when one species benefits, and the other is harmed.


Where in the world are coral reefs found?
Answer: they are frequently found near the drop-offs at the outer edges of the reef,
and less commonly within lagoons.


What species engage in schooling?
Answer: schooling fish have developed remarkable displays of precise choreography whi
ch confuse and evade predators.


Which species are ambush predators?
Answer: another ambush predator is the striated frogfish (right).


Why are some coral reef fish colorful?
Answer: coral reef fishes exhibit a huge variety of dazzling and sometimes bizarre co
lours and patterns.


Which species is blue?
Answer: the psychedelic synchiropus splendidus is one of only two animal species know
n to have blue colouring because of cellular pigment.


What is the slowest species that lives in coral reefs?
Answer: the slowest-moving fishes are the sea horses, often found in reefs.


exit
Thank you for using the Informational Chatbot About Coral Reef Fish. Goodbye.
```

In [30]:

```
chatbot_models = []

for i in range(20):
    chatbot_models.append(1)

model_1_performance_list = ["Incorrect",
                            "Correct",
                            "Correct",
                            "Incorrect",
                            "Incorrect",
                            "Incorrect",
                            "Correct",
                            "Incorrect",
                            "Incorrect",
                            "Incorrect",
                            "Partially Correct",
                            "Correct",
                            "Incorrect",
                            "Incorrect",
                            "Incorrect",
                            "Incorrect",
```

```
                                "Correct",
                                "Incorrect",
                                 "Correct",
                                 "Correct"]
```

2) Chatbot 2 - Fine-Tune GPT2 Model

In [2]:
```
!pip install transformers
```

```
Requirement already satisfied: transformers in c:\users\steve\anaconda3\lib\site-pack
ages (4.31.0)
Requirement already satisfied: filelock in c:\users\steve\anaconda3\lib\site-packages
(from transformers) (3.6.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.14.1 in c:\users\steve\anacond
a3\lib\site-packages (from transformers) (0.16.4)
Requirement already satisfied: numpy>=1.17 in c:\users\steve\anaconda3\lib\site-packa
ges (from transformers) (1.22.4)
Requirement already satisfied: packaging>=20.0 in c:\users\steve\anaconda3\lib\site-p
ackages (from transformers) (21.3)
Requirement already satisfied: pyyaml>=5.1 in c:\users\steve\anaconda3\lib\site-packa
ges (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in c:\users\steve\anaconda3\lib\site
-packages (from transformers) (2022.3.15)
Requirement already satisfied: requests in c:\users\steve\anaconda3\lib\site-packages
(from transformers) (2.31.0)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in c:\users\steve\an
aconda3\lib\site-packages (from transformers) (0.13.3)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\steve\anaconda3\lib\sit
e-packages (from transformers) (0.3.2)
Requirement already satisfied: tqdm>=4.27 in c:\users\steve\anaconda3\lib\site-packag
es (from transformers) (4.66.1)
Requirement already satisfied: fsspec in c:\users\steve\anaconda3\lib\site-packages
(from huggingface-hub<1.0,>=0.14.1->transformers) (2023.6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\steve\anaconda3
\lib\site-packages (from huggingface-hub<1.0,>=0.14.1->transformers) (4.7.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\steve\anaconda3\l
ib\site-packages (from packaging>=20.0->transformers) (3.0.4)
Requirement already satisfied: colorama in c:\users\steve\anaconda3\lib\site-packages
(from tqdm>=4.27->transformers) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\steve\anaconda3\l
ib\site-packages (from requests->transformers) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\steve\anaconda3\lib\site-pack
ages (from requests->transformers) (3.3)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\steve\anaconda3\lib\sit
e-packages (from requests->transformers) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\steve\anaconda3\lib\sit
e-packages (from requests->transformers) (2021.10.8)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\steve\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\steve\anaconda3\lib\site-pa
ckages)
```

In [5]:
```
pip install --upgrade protobuf
```

```
Requirement already satisfied: protobuf in c:\users\steve\anaconda3\lib\site-packages
(3.19.0)
Collecting protobuf
  Obtaining dependency information for protobuf from https://files.pythonhosted.org/p
ackages/13/1f/079588752c462cd0b5e15f6def7d91f0b82489572a86f922b7eab78357a8/protobuf-
4.24.1-cp39-cp39-win_amd64.whl.metadata
  Downloading protobuf-4.24.1-cp39-cp39-win_amd64.whl.metadata (540 bytes)
Downloading protobuf-4.24.1-cp39-cp39-win_amd64.whl (430 kB)
   ---------------------------------------- 430.4/430.4 kB 4.5 MB/s eta 0:00:00
Installing collected packages: protobuf
  Attempting uninstall: protobuf
    Found existing installation: protobuf 3.19.0
    Uninstalling protobuf-3.19.0:
      Successfully uninstalled protobuf-3.19.0
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -rotobuf (c:\users\steve\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\steve\anaconda3\lib\site-pa
ckages)
ERROR: Could not install packages due to an OSError: [WinError 5] Access is denied:
'C:\\Users\\steve\\anaconda3\\Lib\\site-packages\\google\\protobuf\\~-ternal\\_api_im
plementation.cp39-win_amd64.pyd'
Consider using the `--user` option or check the permissions.
```

In [8]:

```python
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer, AdamW

# Load the GPT-2 model and tokenizer
model_name = "gpt2"
model = GPT2LMHeadModel.from_pretrained(model_name)
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
# Define multiple question and answer pairs
qa_pairs = [
    ("Which species are at the top of the food chain in coral reefs?", "sharks and gia
    ("What fish eat coral?", "parrotfish and butterflyfish. STOP"),
    ("Which coral reef fish has the shortest lifespan?", "seven-figure pygmy goby. STO
    ("What species in coral reefs can inflate themselves?", "puffers, striated frogfis
    ("How do sea anemones protect themselves?", "tentacles that bristle tiny harpoons
    ("What species commonly serves as a cleaner fish?", "bluestreak cleaner wrasse. ST
    ("What species in coral reefs eat sponges?", "emperor angelfish. STOP"),
    ("What species in coral reefs eat stingrays?", "caribbean reef shark and great ham
    ("Which species in coral reefs are hermaphrodites", "grouper. STOP"),
    ("What species in coral reefs is known to eat birds?", "blacktip reef shark. STOP'
    ("What percent of the ocean is covered by coral reefs?", "less than 1%. STOP"),
    ("When is the colouration of clown triggerfish vivid?", "when they are not threate
    ("What are reef lizardfish coated in?", "mucus. STOP"),
    ("Name some coral reef fish that feed on zooplankton?", "damselfish, surgeonfish,
    ("Parrotfish often school with what other species?", "spinefoot rabbitfish. STOP")
    ("What species eat sea anemones?", "saddle butterflyfish. STOP"),
    ("How many venomous spines does the reef stonefish have?", "13. STOP"),
    ("What predators eat reef stonefish?", "bottom feeding rays, sharks, and Stokes' s
    ("What toxin is sometimes found coral reef carnivores?", "ciguatera toxin. STOP"),
    ("How long are whitetip reef sharks?", "usually less than 1.6 meters. STOP"),
    ("How long are whitetip reef sharks?", "less than 1.6 meters. STOP"),
    ("What predators eat whitetip reef sharks?", "tiger sharks and Galapagos sharks. S
    ("What predators eat blacktip reef sharks?", "groupers, grey reef sharks, tiger sh
    ("What sharks are known for having small home ranges?", "blacktip reef sharks. STO
    ("Are grey reef sharks social or territorial?", "social. STOP"),
    ("Name a coral reef shark that likes drop-offs?", "whitetip reef shark, blacktip r
```

```
    ("What do benthic algae grow on?", "dead coral and other inert surfaces. STOP"),
    ("What do goatfish use to search for food?", "chemosensory barbels (whiskers). STO
    ("Goatfish commonly change their colouration to match that of which fish?", "snapp
    ("What does the tassled scorpionfish camouflage itself to look like?", "coral encr
    ("What are some threats to the survival of coral reef fish?", "pollution, overfish
    ("What percent of marine fish species live in coral reefs?", "25 percent. STOP"),
    ("Where do most coral reef fish have spines?", "on their fins. STOP"),
    ("Why do toadfish sing?", "to attract mates. STOP"),
    ("What do Synchiropus splendidus eat?", "small crustaceans and other invertebrates
    ("What color is the mouth of the clown triggerfish?", "bright yellow. STOP"),
    ("How many species of parasites would go extinct if one coral reef fish species of
    ("What types of sharks can enter brackish or freshwater environments?", "blacktip
    ("What sound frequency do whitetip reef sharks recognize as coming from struggling
    ("Can whitetip reef sharks rest on the sea floor or do they have to keep moving?",
    ("Are whitetip reef sharks better at hunting in tight crevices or in open water?",
    ("How long are Caribbean reef sharks?", "up to 3 meters. STOP"),
    ("Titan triggerfish use jets of water to expose what species buried in the sand?",
    ("Are coral reef fish bodies often optimized for straight-line speed or for manoeu
    ("Where does the foureye butterflyfish get its name from?", "the large dark spots
    ("How long are striated frogfish?", "about 10 centimeters. STOP"),
    ("Why do some coral reef fish engage in schooling?", "defense against predators th
    ("What are lateral lines in coral reef fish?", "pressure sensors that allow school
    ("What is a primary producer?", "a plant that synthesizes food from solar energy.
    ("What is the name for the stinging cells in fire coral?", " nematocysts. STOP")
]

# Concatenate the question and answer pairs with appropriate formatting
formatted_pairs = [f"Q: {q}\nA: {a}\n" for q, a in qa_pairs]
qa_text = "\n".join(formatted_pairs)

# Fine-tune the GPT-2 model with the Q&A pairs
inputs = tokenizer.encode(qa_text, return_tensors="pt", max_length = 1000)
model.train()

# Define the optimizer
optimizer = AdamW(model.parameters(), lr=1e-5)

# Run the fine-tuning loop (example: 1 epoch)
for j in range(120):
    print (j)
    outputs = model(inputs, labels=inputs)
    loss = outputs.loss
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()

# Save the fine-tuned model
model.save_pretrained("fine-tuned-gpt2")

# Load the fine-tuned model
fine_tuned_model = GPT2LMHeadModel.from_pretrained("fine-tuned-gpt2")
```

```
Truncation was not explicitly activated but `max_length` is provided a specific valu
e, please use `truncation=True` to explicitly truncate examples to max length. Defaul
ting to 'longest_first' truncation strategy. If you encode pairs of sequences (GLUE-s
tyle) with the tokenizer you can select this strategy more precisely by providing a s
pecific strategy to `truncation`.
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
```

In [9]:
```
# Test the fine-tuned model with a question
test_question = "Q: How many species of fish live in coral reefs?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: How many species of fish live in coral reefs?', 'A: About 1,000 species of coral
reef fish live in coral reefs.']
```

In [10]:
```
# Test the fine-tuned model with a question
test_question = "Q: What is the most venomous fish?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What is the most venomous fish?', 'A: blacktip reef shark.']
```

In [11]:
```
# Test the fine-tuned model with a question
test_question = "Q: What sharks live in coral reefs?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What sharks live in coral reefs?', 'A: sharks. Sharks are the most abundant mari
ne invertebrates on Earth. They are the largest marine invertebrates on Earth and are
responsible for over half of all marine invertebrates. Sharks are also the most abund
ant marine invertebrates on Earth and are responsible for over half of all marine inv
ertebrates. Sharks are also the most abundant marine invertebrates on Earth and are r
esponsible for over half of all marine invertebrates.']
```

In [12]:
```
# Test the fine-tuned model with a question
test_question = "Q: What fish are poisonous?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What fish are poisonous?', 'A: blacktip reef shark and blacktip reef shark.']
```

In [13]:
```
# Test the fine-tuned model with a question
test_question = "Q: What fish can electrocute you?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What fish can electrocute you?', 'A: blacktip reef shark']
```

In [14]:
```
# Test the fine-tuned model with a question
test_question = "Q: What species can sting you?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What species can sting you?', 'A: sharks and rays.']
```

In [15]:
```
# Test the fine-tuned model with a question
test_question = "Q: What species are parasitic?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
```

```
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What species are parasitic?', 'A: scorpionfish.']
```

In [16]:
```
# Test the fine-tuned model with a question
test_question = "Q: What species are venomous?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What species are venomous?', 'A: scorpionfish.']
```

In [17]:
```
# Test the fine-tuned model with a question
test_question = "Q: What species are known for attacking scuba divers?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What species are known for attacking scuba divers?', 'A: sharks, rays, and other
sharks.']
```

In [18]:
```
# Test the fine-tuned model with a question
test_question = "Q: What are common herbivorous fish?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)
```

```
print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What are common herbivorous fish?', "A: Stokes' sea bass. Stokes' sea bass are c
arnivorous fish that eat fish that are not their own. Stokes' sea bass are carnivorou
s fish that eat fish that are not their own. Stokes' sea bass are carnivorous fish th
at eat fish that are not their own."]
```

In [19]:
```
# Test the fine-tuned model with a question
test_question = "Q: Why do fish camouflage themselves?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: Why do fish camouflage themselves?', 'A: Because they are predators.']
```

In [20]:
```
# Test the fine-tuned model with a question
test_question = "Q: How do fish get rid of their parasites?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: How do fish get rid of their parasites?', 'A: They use parasites to attract mate
s.']
```

In [21]:
```
# Test the fine-tuned model with a question
test_question = "Q: What species have a mutualistic relationship?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)
```

```
print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What species have a mutualistic relationship?', 'A: sharks and sea urchins.']
```

In [22]:
```
# Test the fine-tuned model with a question
test_question = "Q: What species have a commensalistic relationship?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What species have a commensalistic relationship?', 'A: giant butterfly and butte
rfly.']
```

In [23]:
```
# Test the fine-tuned model with a question
test_question = "Q: Where in the world are coral reefs found?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: Where in the world are coral reefs found?', 'A: coral reefs are found in tropica
l and subtropical environments. They are the most abundant coral reef species in the
world.']
```

In [24]:
```
# Test the fine-tuned model with a question
test_question = "Q: What species engage in schooling?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What species engage in schooling?', 'A: sharks, which are the most common predat
or of sharks.']
```

In [30]:
```
# Test the fine-tuned model with a question
test_question = "Q: What species are ambush predators?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What species are ambush predators?', 'A: sharks and giant moray. ']
```

In [25]:
```
# Test the fine-tuned model with a question
test_question = "Q: Which species are ambush predators?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: Which species are ambush predators?', 'A: giant frogfish.']
```

In [26]:
```
# Test the fine-tuned model with a question
test_question = "Q: Why are some coral reef fish colorful?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
```

```
Generated completion for the test question:
['Q: Why are some coral reef fish colorful?', 'A: Because they are not as bright as o
ther reef fish.']
```

In [27]:
```python
# Test the fine-tuned model with a question
test_question = "Q: Which species is blue?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: Which species is blue?', 'A: blackfish.']
```

In [28]:
```python
# Test the fine-tuned model with a question
test_question = "Q: What is the slowest species that lives in coral reefs?"

# Generate a completion for the test question
input_ids = tokenizer.encode(test_question, return_tensors="pt")
output = fine_tuned_model.generate(input_ids, max_length=100)
completion = tokenizer.decode(output[0], skip_special_tokens=True)

print("Generated completion for the test question:")
print(completion.split("STOP")[0].splitlines()[0:2])
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
ve unexpected behavior. Please pass your input's `attention_mask` to obtain reliable
results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated completion for the test question:
['Q: What is the slowest species that lives in coral reefs?', 'A: sharks.']
```

In [31]:
```python
for i in range(20):
    chatbot_models.append(2)


model_2_performance_list = ["Partially Correct",
                            "Incorrect",
                            "Incorrect",
                            "Incorrect",
                            "Incorrect",
                            "Partially Correct",
                            "Incorrect",
                            "Correct",
                            "Partially Correct",
                            "Incorrect",
                            "Partially Correct",
                            "Incorrect",
                            "Incorrect",
                            "Incorrect",
                            "Correct",
                            "Incorrect",
```

```
                           "Correct",
                           "Incorrect",
                           "Incorrect",
                           "Incorrect"]
```

3) Chatbot 3 - Chatbot Emphasizing Cosine Similarity of TF-IDF Representations of Sententces

In [35]:
```
import io
import random
import string # to process standard python strings
import warnings
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import warnings
warnings.filterwarnings('ignore')
```

In [36]:
```
import nltk
from nltk.stem import WordNetLemmatizer
nltk.download('popular', quiet=True) # for downloading packages
#nltk.download('punkt') # first-time use only
#nltk.download('wordnet') # first-time use only
```

Out[36]:
```
True
```

In [37]:
```
CORPUS_PATH = 'C:/Users/steve/OneDrive/Desktop/Github/Natural_Language_Processing/Chat
f=open(CORPUS_PATH,'r',errors = 'ignore')
raw=f.read()
raw = raw.lower()# converts to lowercase
```

In [38]:
```
sent_tokens = nltk.sent_tokenize(raw)# converts to list of sentences
word_tokens = nltk.word_tokenize(raw)# converts to list of words
```

In [39]:
```
lemmer = nltk.stem.WordNetLemmatizer()
#WordNet is a semantically-oriented dictionary of English included in NLTK.
def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]
remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)

def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

In [40]:
```
GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's up","hey",)
GREETING_RESPONSES = ["Hi", "Hey", "*nods*", "Hi there", "Hello", "I am glad! You are
def greeting(sentence):

    for word in sentence.split():
        if word.lower() in GREETING_INPUTS:
            return random.choice(GREETING_RESPONSES)
```

In [41]:
```
def response(user_response):
    robo_response=''
    sent_tokens.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
```

```
    tfidf = TfidfVec.fit_transform(sent_tokens)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens[idx]
        return robo_response
```

In [42]:

```
flag=True
print("Hello, I am the Third Coral Reef Chatbot! I will answer your queries related to
while(flag==True):
    user_response = input()
    user_response=user_response.lower()
    if(user_response!='bye'):
        if(user_response=='thanks' or user_response=='thank you' ):
            flag=False
            print("You are welcome.")
        else:
            if(greeting(user_response)!=None):
                print(greeting(user_response))
            else:
                print("",end="")
                print(response(user_response))
                sent_tokens.remove(user_response)
    else:
        flag=False
        print("Bye! take care..")
```

```
Hello, I am the Third Coral Reef Chatbot! I will answer your queries related to coral
reef fish. If you want to exit, type Bye!
How many species of fish live in coral reefs?
 coral reef fish are fish which live amongst or in close relation to coral reefs.
What is the most venomous fish?
 there is a distinction between poisonous fish and venomous fish.
What sharks live in coral reefs?
 coral reef fish are fish which live amongst or in close relation to coral reefs.
What fish are poisonous?
 there is a distinction between poisonous fish and venomous fish.
What fish can electrocute you?
 as with all fish, coral reef fish harbour parasites.
What species can sting you?
 in return, the anemones provide the clownfish protection from their predators, who a
re not immune to anemone stings.
What species are venomous?
 the most venomous known fish is the reef stonefish.
What species are parasitic?
 parasites of coral reef fish include nematodes, platyhelminthes (cestodes, digenean
s, and monogeneans), leeches, parasitic crustaceans such as isopods and copepods, and
various microorganisms such as myxosporidia and microsporidia.
What species are known for attacking scuba divers?
 lionfish can aggressively dart at scuba divers and attempt to puncture the facemask
with their venomous spines.
What are common herbivorous fish?
 sea anemones are common on reefs.
Why do fish camouflage themselves?
 sometimes they camouflage the fish when the fish rests in places with the right back
ground.
How do fish get rid of their parasites?
 as with all fish, coral reef fish harbour parasites.
What species have mutualistic relationships?
 there is a mutualistic relationship between sea anemones and clownfish.
What species have commensalistic relationships?
 the relationship can be mutualistic, when both species benefit from the relationshi
p, commensalistic, when one species benefits and the other is unaffected, and parasit
istic, when one species benefits, and the other is harmed.
Where in the world are coral reefs found?
 coral reefs contain the most diverse fish assemblages to be found anywhere on earth,
with perhaps as many as 6,000â€“8,000 species dwelling within coral reef ecosystems o
f the world's oceans.
What species engage in schooling?
 they have evolved to find protection by schooling, sometimes with other species like
shoaling rabbitfish.
What species are ambush predators?
 the well camouflaged striated frogfish, a species of anglerfish, is an ambush predat
or.
Why are some coral reef fish colorful?
 coral reef fish are fish which live amongst or in close relation to coral reefs.
What species is blue?
 the psychedelic synchiropus splendidus is one of only two animal species known to ha
ve blue colouring because of cellular pigment.
What is the slowest species that lives in coral reefs?
 some of these fish parasites have heteroxenous life cycles (i.e.
exit
 after dusk, a group of sharks may target the same prey item, covering every exit rou
te from a particular coral head.
bye
Bye! take care..
```

In [32]:
```
for i in range(20):
    chatbot_models.append(3)

model_3_performance_list = ["Incorrect",
                            "Incorrect",
                            "Incorrect",
                            "Incorrect",
                            "Incorrect",
                            "Correct",
                            "Correct",
                            "Correct",
                            "Correct",
                            "Incorrect",
                            "Partially Correct",
                            "Incorrect",
                            "Correct",
                            "Incorrect",
                            "Partially Correct",
                            "Correct",
                            "Correct",
                            "Incorrect",
                             "Correct",
                             "Incorrect"]
```

4) Chatbot 4 - Distillbert Chatbot

Source: https://huggingface.co/distilbert-base-cased-distilled-squad

In [4]:
```
#read in data
CORPUS_PATH = 'C:/Users/steve/OneDrive/Desktop/Github/Natural_Language_Processing/Chat
f=open(CORPUS_PATH,'r',errors = 'ignore')
raw=f.read()
raw=raw.lower()# converts to lowercase
```

In [8]:
```
from transformers import pipeline
question_answerer = pipeline("question-answering", model='distilbert-base-cased-distil

#context = r"""
#Extractive Question Answering is the task of extracting an answer from a text given a
#question answering dataset is the SQuAD dataset, which is entirely based on that task
#a model on a SQuAD task, you may leverage the examples/pytorch/question-answering/run
#"""

result = question_answerer(question="What is a good example of a question answering da
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu

#Answer: 'SQuAD dataset', score: 0.5152, start: 147, end: 160
```

```
Downloading (…)lve/main/config.json:   0%|          | 0.00/473 [00:00<?, ?B/s]
```

```
C:\Users\steve\anaconda3\lib\site-packages\huggingface_hub\file_download.py:133: User
Warning: `huggingface_hub` cache-system uses symlinks by default to efficiently store
duplicated files but your machine does not support them in C:\Users\steve\.cache\hugg
ingface\hub. Caching files will still work but in a degraded version that might requi
re more space on your disk. This warning can be disabled by setting the `HF_HUB_DISAB
LE_SYMLINKS_WARNING` environment variable. For more details, see https://huggingface.
co/docs/huggingface_hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run
Python as an administrator. In order to see activate developer mode, see this articl
e: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-d
evelopment
  warnings.warn(message)
Downloading model.safetensors:   0%|          | 0.00/261M [00:00<?, ?B/s]
Downloading (…)okenizer_config.json:   0%|          | 0.00/29.0 [00:00<?, ?B/s]
Downloading (…)solve/main/vocab.txt:   0%|          | 0.00/213k [00:00<?, ?B/s]
Downloading (…)/main/tokenizer.json:   0%|          | 0.00/436k [00:00<?, ?B/s]
Answer: 'the great hammerhead uses its hammer', score: 0.0377, start: 34598, end: 346
34
```

In [9]:
```
result = question_answerer(question="How many species of fish live in coral reefs?",
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'four', score: 0.6391, start: 17258, end: 17262
```

In [10]:
```
result = question_answerer(question="What is the most venomous fish?", context=raw)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'reef stonefish', score: 0.8257, start: 24899, end: 24913
```

In [11]:
```
result = question_answerer(question="What sharks live in coral reefs?",     context=ra
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'whitetip reef shark', score: 0.8656, start: 34283, end: 34302
```

In [12]:
```
result = question_answerer(question="What fish are poisonous?",     context=raw)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'coral reef fish', score: 0.8148, start: 0, end: 15
```

In [13]:
```
result = question_answerer(question="What fish can electrocute you?",     context=raw)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'snapper', score: 0.3421, start: 11139, end: 11146
```

In [14]:
```
result = question_answerer(question="What species can sting you?",     context=raw)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'yellow stingrays', score: 0.6958, start: 34108, end: 34124
```

In [15]:
```
result = question_answerer(question="What species are parasitic?", context=raw)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'coral reef fish', score: 0.5431, start: 0, end: 15
```

In [16]:
```
result = question_answerer(question="What species are venomous?", context=raw)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'spinefoot rabbitfish', score: 0.9286, start: 18449, end: 18469
```

In [17]:
```
result = question_answerer(question="What species are known for attacking scuba divers
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'lionfish', score: 0.9109, start: 26369, end: 26377
```

In [18]:
```
result = question_answerer(question="What are common herbivorous fish?", context=raw)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'yellow tang', score: 0.5247, start: 2116, end: 2127
```

In [19]:
```
result = question_answerer(question="Why do fish camouflage themselves?", context=raw)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'when the fish rests in places with the right background', score: 0.7476, sta
rt: 6715, end: 6770
```

In [20]:
```
result = question_answerer(question="How do fish get rid of their parasites?", context
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'by biting, stinging, or stabbing', score: 0.4523, start: 23987, end: 24019
```

In [21]:
```
result = question_answerer(question="What species have mutualistic relationships?", co
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'sea anemones and clownfish', score: 0.8657, start: 21835, end: 21861
```

In [22]:
```
result = question_answerer(question="What species have commensalistic relationships?",
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'sea anemones and clownfish', score: 0.6863, start: 21835, end: 21861
```

In [23]:
```
result = question_answerer(question="Where in the world are coral reefs found?", conte
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'top predators', score: 0.8029, start: 26812, end: 26825
```

In [24]:
```
result = question_answerer(question="What species engage in schooling?", context=raw)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'spinefoot rabbitfish', score: 0.6337, start: 19619, end: 19639
```

In [25]:
```
result = question_answerer(question="Which species are ambush predators?", context=raw
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'frogfish', score: 0.5638, start: 9688, end: 9696
```

In [26]:
```
result = question_answerer(question="Why are some coral reef fish colorful?", context=
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'high biodiversity', score: 0.7926, start: 22542, end: 22559
```

In [27]:
```
result = question_answerer(question="Which species is blue?", context=raw)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'bluestripe snapper', score: 0.6375, start: 12029, end: 12047
```

In [28]:
```
result = question_answerer(question="What is the slowest species that lives in coral r
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {resu
```

```
Answer: 'dwarf seahorse', score: 0.4608, start: 4939, end: 4953
```

In [33]:
```
for i in range(20):
    chatbot_models.append(4)

model_4_performance_list = ["Incorrect",
                            "Correct",
```

```
                            "Correct",
                            "Incorrect",
                            "Incorrect",
                            "Correct",
                            "Incorrect",
                            "Correct",
                            "Correct",
                            "Correct",
                            "Incorrect",
                            "Incorrect",
                            "Correct",
                            "Incorrect",
                            "Incorrect",
                            "Correct",
                            "Correct",
                            "Incorrect",
                            "Partially Correct",
                            "Correct"]
```

5) Chatbot 5 - Roberta Chatbot

https://huggingface.co/deepset/roberta-base-squad2

In [46]:
```
from transformers import AutoModelForQuestionAnswering, AutoTokenizer, pipeline, Rober
from transformers import optimization

batch_size = 96
n_epochs = 2
base_LM_model = "roberta-base"
max_seq_len = 386
learning_rate = 3e-5
#lr_schedule = optimization.LinearWarmup
warmup_proportion = 0.2
doc_stride=128
max_query_length=64
```

In [50]:
```
#!pip install farm-haystack[inference]
```

```
Collecting farm-haystack[inference]
  Obtaining dependency information for farm-haystack[inference] from https://files.py
thonhosted.org/packages/df/ef/485cd648ee02afafd5c014b609c214299507112c246b75303f91fd2
c139f/farm_haystack-1.19.0-py3-none-any.whl.metadata
  Downloading farm_haystack-1.19.0-py3-none-any.whl.metadata (25 kB)
Collecting boilerpy3 (from farm-haystack[inference])
  Downloading boilerpy3-1.0.6-py3-none-any.whl (22 kB)
Collecting canals==0.3.2 (from farm-haystack[inference])
  Obtaining dependency information for canals==0.3.2 from https://files.pythonhosted.
org/packages/b8/f6/6d2071a20400129a72390f021b46603f694d29553df0725152864c3c40f3/canal
s-0.3.2-py3-none-any.whl.metadata
  Downloading canals-0.3.2-py3-none-any.whl.metadata (4.4 kB)
Collecting events (from farm-haystack[inference])
  Obtaining dependency information for events from https://files.pythonhosted.org/pac
kages/25/ed/e47dec0626edd468c84c04d97769e7ab4ea6457b7f54dcb3f72b17fcd876/Events-0.5-p
y3-none-any.whl.metadata
  Downloading Events-0.5-py3-none-any.whl.metadata (3.9 kB)
Requirement already satisfied: jsonschema in c:\users\steve\anaconda3\lib\site-packag
es (from farm-haystack[inference]) (4.4.0)
Collecting lazy-imports==0.3.1 (from farm-haystack[inference])
  Downloading lazy_imports-0.3.1-py3-none-any.whl (12 kB)
Collecting more-itertools (from farm-haystack[inference])
  Obtaining dependency information for more-itertools from https://files.pythonhoste
d.org/packages/5a/cb/6dce742ea14e47d6f565589e859ad225f2a5de576d7696e0623b784e226b/mor
e_itertools-10.1.0-py3-none-any.whl.metadata
  Downloading more_itertools-10.1.0-py3-none-any.whl.metadata (33 kB)
Requirement already satisfied: networkx in c:\users\steve\anaconda3\lib\site-packages
(from farm-haystack[inference]) (2.7.1)
Requirement already satisfied: pandas in c:\users\steve\anaconda3\lib\site-packages
(from farm-haystack[inference]) (1.4.2)
Requirement already satisfied: pillow in c:\users\steve\anaconda3\lib\site-packages
(from farm-haystack[inference]) (9.0.1)
Collecting platformdirs (from farm-haystack[inference])
  Obtaining dependency information for platformdirs from https://files.pythonhosted.o
rg/packages/14/51/fe5a0d6ea589f0d4a1b97824fb518962ad48b27cd346dcdfa2405187997a/platfo
rmdirs-3.10.0-py3-none-any.whl.metadata
  Downloading platformdirs-3.10.0-py3-none-any.whl.metadata (11 kB)
Collecting posthog (from farm-haystack[inference])
  Obtaining dependency information for posthog from https://files.pythonhosted.org/pa
ckages/a7/73/35758818228c70348be4c3c66a76653c62e894e0e3c3461453c5341ca926/posthog-3.
0.2-py2.py3-none-any.whl.metadata
  Downloading posthog-3.0.2-py2.py3-none-any.whl.metadata (2.0 kB)
Collecting prompthub-py==4.0.0 (from farm-haystack[inference])
  Obtaining dependency information for prompthub-py==4.0.0 from https://files.pythonh
osted.org/packages/27/5f/8c4939e290ff93af79364b88ffe3902d29c234f94e8227cf0b7fce3c887
f/prompthub_py-4.0.0-py3-none-any.whl.metadata
  Downloading prompthub_py-4.0.0-py3-none-any.whl.metadata (2.2 kB)
Requirement already satisfied: pydantic<2 in c:\users\steve\anaconda3\lib\site-packag
es (from farm-haystack[inference]) (1.10.12)
Collecting quantulum3 (from farm-haystack[inference])
  Downloading quantulum3-0.9.0-py3-none-any.whl (10.7 MB)
     ---------------------------------------- 10.7/10.7 MB 11.1 MB/s eta 0:00:00
Collecting rank-bm25 (from farm-haystack[inference])
  Downloading rank_bm25-0.2.2-py3-none-any.whl (8.6 kB)
Requirement already satisfied: requests in c:\users\steve\anaconda3\lib\site-packages
(from farm-haystack[inference]) (2.27.1)
Collecting requests-cache<1.0.0 (from farm-haystack[inference])
  Downloading requests_cache-0.9.8-py3-none-any.whl (48 kB)
```

```
WARNING: Ignoring invalid distribution -rotobuf (c:\users\steve\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\steve\anaconda3\lib\site-pa
ckages)
ERROR: pip's dependency resolver does not currently take into account all the package
s that are installed. This behaviour is the source of the following dependency confli
cts.
conda-repo-cli 1.0.4 requires pathlib, which is not installed.
spyder 5.1.5 requires pyqt5<5.13, which is not installed.
spyder 5.1.5 requires pyqtwebengine<5.13, which is not installed.
google-api-core 1.25.1 requires google-auth<2.0dev,>=1.21.1, but you have google-auth
2.18.1 which is incompatible.
google-cloud-core 1.7.1 requires google-auth<2.0dev,>=1.24.0, but you have google-aut
h 2.18.1 which is incompatible.
google-cloud-storage 1.31.0 requires google-auth<2.0dev,>=1.11.0, but you have google
-auth 2.18.1 which is incompatible.
tensorboard 2.12.3 requires tensorboard-data-server<0.8.0,>=0.7.0, but you have tenso
rboard-data-server 0.6.1 which is incompatible.
tensorflow-intel 2.12.0 requires protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.
4,!=4.21.5,<5.0.0dev,>=3.20.3, but you have protobuf 4.21.0 which is incompatible.
```

```
     ---------------------------------------- 48.7/48.7 kB ? eta 0:00:00
Collecting scikit-learn>=1.3.0 (from farm-haystack[inference])
  Obtaining dependency information for scikit-learn>=1.3.0 from https://files.pythonh
osted.org/packages/2d/30/3afb8bcb785653254eb646ff2680ec4d637b40b06f4b046aca17b5e086b
0/scikit_learn-1.3.0-cp39-cp39-win_amd64.whl.metadata
  Downloading scikit_learn-1.3.0-cp39-cp39-win_amd64.whl.metadata (11 kB)
Collecting sseclient-py (from farm-haystack[inference])
  Downloading sseclient_py-1.7.2-py2.py3-none-any.whl (8.4 kB)
Requirement already satisfied: tenacity in c:\users\steve\anaconda3\lib\site-packages
(from farm-haystack[inference]) (8.0.1)
Collecting tiktoken>=0.3.2 (from farm-haystack[inference])
  Downloading tiktoken-0.4.0-cp39-cp39-win_amd64.whl (635 kB)
     ---------------------------------------- 635.6/635.6 kB 8.0 MB/s eta 0:00:00
Requirement already satisfied: tqdm in c:\users\steve\anaconda3\lib\site-packages (fr
om farm-haystack[inference]) (4.64.0)
Requirement already satisfied: transformers==4.31.0 in c:\users\steve\anaconda3\lib\s
ite-packages (from farm-haystack[inference]) (4.31.0)
Requirement already satisfied: huggingface-hub>=0.5.0 in c:\users\steve\anaconda3\lib
\site-packages (from farm-haystack[inference]) (0.16.4)
Requirement already satisfied: sentence-transformers>=2.2.0 in c:\users\steve\anacond
a3\lib\site-packages (from farm-haystack[inference]) (2.2.2)
Requirement already satisfied: pyyaml<7.0,>=6.0 in c:\users\steve\anaconda3\lib\site-
packages (from prompthub-py==4.0.0->farm-haystack[inference]) (6.0)
Collecting requests (from farm-haystack[inference])
  Obtaining dependency information for requests from https://files.pythonhosted.org/p
ackages/70/8e/0e2d847013cb52cd35b38c009bb167a1a26b2ce6cd6965bf26b47bc0bf44/requests-
2.31.0-py3-none-any.whl.metadata
  Downloading requests-2.31.0-py3-none-any.whl.metadata (4.6 kB)
Requirement already satisfied: filelock in c:\users\steve\anaconda3\lib\site-packages
(from transformers==4.31.0->farm-haystack[inference]) (3.6.0)
Requirement already satisfied: numpy>=1.17 in c:\users\steve\anaconda3\lib\site-packa
ges (from transformers==4.31.0->farm-haystack[inference]) (1.22.4)
Requirement already satisfied: packaging>=20.0 in c:\users\steve\anaconda3\lib\site-p
ackages (from transformers==4.31.0->farm-haystack[inference]) (21.3)
Requirement already satisfied: regex!=2019.12.17 in c:\users\steve\anaconda3\lib\site
-packages (from transformers==4.31.0->farm-haystack[inference]) (2022.3.15)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in c:\users\steve\an
aconda3\lib\site-packages (from transformers==4.31.0->farm-haystack[inference]) (0.1
3.3)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\steve\anaconda3\lib\sit
e-packages (from transformers==4.31.0->farm-haystack[inference]) (0.3.2)
Requirement already satisfied: sentencepiece!=0.1.92,>=0.1.91 in c:\users\steve\anaco
nda3\lib\site-packages (from transformers==4.31.0->farm-haystack[inference]) (0.1.99)
Requirement already satisfied: protobuf in c:\users\steve\anaconda3\lib\site-packages
(from transformers==4.31.0->farm-haystack[inference]) (4.21.0)
Requirement already satisfied: torch!=1.12.0,>=1.9 in c:\users\steve\anaconda3\lib\si
te-packages (from transformers==4.31.0->farm-haystack[inference]) (2.0.1)
Collecting accelerate>=0.20.3 (from transformers==4.31.0->farm-haystack[inference])
  Obtaining dependency information for accelerate>=0.20.3 from https://files.pythonho
sted.org/packages/4d/a7/05c67003d659a0035f2b3a8cf389c1d9645865aee84a73ce99ddab16682f/
accelerate-0.22.0-py3-none-any.whl.metadata
  Downloading accelerate-0.22.0-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: fsspec in c:\users\steve\anaconda3\lib\site-packages
(from huggingface-hub>=0.5.0->farm-haystack[inference]) (2022.2.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\steve\anaconda3
\lib\site-packages (from huggingface-hub>=0.5.0->farm-haystack[inference]) (4.7.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\steve\anaconda3\l
ib\site-packages (from requests->farm-haystack[inference]) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\steve\anaconda3\lib\site-pack
ages (from requests->farm-haystack[inference]) (3.3)
```

```
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\steve\anaconda3\lib\sit
e-packages (from requests->farm-haystack[inference]) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\steve\anaconda3\lib\sit
e-packages (from requests->farm-haystack[inference]) (2021.10.8)
Requirement already satisfied: appdirs>=1.4.4 in c:\users\steve\anaconda3\lib\site-pa
ckages (from requests-cache<1.0.0->farm-haystack[inference]) (1.4.4)
Requirement already satisfied: attrs>=21.2 in c:\users\steve\anaconda3\lib\site-packa
ges (from requests-cache<1.0.0->farm-haystack[inference]) (21.4.0)
Collecting cattrs>=22.2 (from requests-cache<1.0.0->farm-haystack[inference])
  Obtaining dependency information for cattrs>=22.2 from https://files.pythonhosted.o
rg/packages/3a/ba/05df14efaa0624fac6b1510e87f5ce446208d2f6ce50270a89b6268aebfe/cattrs
-23.1.2-py3-none-any.whl.metadata
  Downloading cattrs-23.1.2-py3-none-any.whl.metadata (9.3 kB)
Collecting url-normalize>=1.4 (from requests-cache<1.0.0->farm-haystack[inference])
  Downloading url_normalize-1.4.3-py2.py3-none-any.whl (6.8 kB)
Requirement already satisfied: scipy>=1.5.0 in c:\users\steve\anaconda3\lib\site-pack
ages (from scikit-learn>=1.3.0->farm-haystack[inference]) (1.7.3)
Collecting joblib>=1.1.1 (from scikit-learn>=1.3.0->farm-haystack[inference])
  Obtaining dependency information for joblib>=1.1.1 from https://files.pythonhosted.
org/packages/10/40/d551139c85db202f1f384ba8bcf96aca2f329440a844f924c8a0040b6d02/jobli
b-1.3.2-py3-none-any.whl.metadata
  Downloading joblib-1.3.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\steve\anaconda3\lib\s
ite-packages (from scikit-learn>=1.3.0->farm-haystack[inference]) (2.2.0)
Requirement already satisfied: torchvision in c:\users\steve\anaconda3\lib\site-packa
ges (from sentence-transformers>=2.2.0->farm-haystack[inference]) (0.15.2)
Requirement already satisfied: nltk in c:\users\steve\anaconda3\lib\site-packages (fr
om sentence-transformers>=2.2.0->farm-haystack[inference]) (3.7)
Requirement already satisfied: colorama in c:\users\steve\anaconda3\lib\site-packages
(from tqdm->farm-haystack[inference]) (0.4.6)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\us
ers\steve\anaconda3\lib\site-packages (from jsonschema->farm-haystack[inference]) (0.
18.0)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\steve\anaconda3\lib
\site-packages (from pandas->farm-haystack[inference]) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\steve\anaconda3\lib\site-pack
ages (from pandas->farm-haystack[inference]) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\steve\anaconda3\lib\site-packages
(from posthog->farm-haystack[inference]) (1.16.0)
Collecting monotonic>=1.5 (from posthog->farm-haystack[inference])
  Downloading monotonic-1.6-py2.py3-none-any.whl (8.2 kB)
Collecting backoff>=1.10.0 (from posthog->farm-haystack[inference])
  Downloading backoff-2.2.1-py3-none-any.whl (15 kB)
Collecting inflect (from quantulum3->farm-haystack[inference])
  Obtaining dependency information for inflect from https://files.pythonhosted.org/pa
ckages/fb/c6/d9feb758be584f729424390af24687d3a4363d968164f94079f83cd536b4/inflect-7.
0.0-py3-none-any.whl.metadata
  Downloading inflect-7.0.0-py3-none-any.whl.metadata (21 kB)
Collecting num2words (from quantulum3->farm-haystack[inference])
  Downloading num2words-0.5.12-py3-none-any.whl (125 kB)
     ---------------------------------------- 125.2/125.2 kB ? eta 0:00:00
Requirement already satisfied: psutil in c:\users\steve\anaconda3\lib\site-packages
(from accelerate>=0.20.3->transformers==4.31.0->farm-haystack[inference]) (5.8.0)
Collecting exceptiongroup (from cattrs>=22.2->requests-cache<1.0.0->farm-haystack[inf
erence])
  Obtaining dependency information for exceptiongroup from https://files.pythonhoste
d.org/packages/ad/83/b71e58666f156a39fb29417e4c8ca4bc7400c0dd4ed9e8842ab54dc8c344/exc
eptiongroup-1.1.3-py3-none-any.whl.metadata
  Downloading exceptiongroup-1.1.3-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\steve\anaconda3\l
```

```
ib\site-packages (from packaging>=20.0->transformers==4.31.0->farm-haystack[inferenc
e]) (3.0.4)
Requirement already satisfied: sympy in c:\users\steve\anaconda3\lib\site-packages (f
rom torch!=1.12.0,>=1.9->transformers==4.31.0->farm-haystack[inference]) (1.10.1)
Requirement already satisfied: jinja2 in c:\users\steve\anaconda3\lib\site-packages
(from torch!=1.12.0,>=1.9->transformers==4.31.0->farm-haystack[inference]) (2.11.3)
Requirement already satisfied: click in c:\users\steve\anaconda3\lib\site-packages (f
rom nltk->sentence-transformers>=2.2.0->farm-haystack[inference]) (8.0.4)
Collecting docopt>=0.6.2 (from num2words->quantulum3->farm-haystack[inference])
  Downloading docopt-0.6.2.tar.gz (25 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\steve\anaconda3\lib\site-
packages (from jinja2->torch!=1.12.0,>=1.9->transformers==4.31.0->farm-haystack[infer
ence]) (2.0.1)
Requirement already satisfied: mpmath>=0.19 in c:\users\steve\anaconda3\lib\site-pack
ages (from sympy->torch!=1.12.0,>=1.9->transformers==4.31.0->farm-haystack[inferenc
e]) (1.2.1)
Downloading canals-0.3.2-py3-none-any.whl (34 kB)
Downloading prompthub_py-4.0.0-py3-none-any.whl (6.9 kB)
Downloading requests-2.31.0-py3-none-any.whl (62 kB)
   ---------------------------------------- 62.6/62.6 kB 3.3 MB/s eta 0:00:00
Downloading scikit_learn-1.3.0-cp39-cp39-win_amd64.whl (9.3 MB)
   ---------------------------------------- 9.3/9.3 MB 35.0 MB/s eta 0:00:00
Downloading Events-0.5-py3-none-any.whl (6.8 kB)
Downloading farm_haystack-1.19.0-py3-none-any.whl (764 kB)
   ---------------------------------------- 764.0/764.0 kB 24.3 MB/s eta 0:00:00
Downloading more_itertools-10.1.0-py3-none-any.whl (55 kB)
   ---------------------------------------- 55.8/55.8 kB ? eta 0:00:00
Downloading platformdirs-3.10.0-py3-none-any.whl (17 kB)
Downloading posthog-3.0.2-py2.py3-none-any.whl (37 kB)
Downloading accelerate-0.22.0-py3-none-any.whl (251 kB)
   ---------------------------------------- 251.2/251.2 kB 15.1 MB/s eta 0:00:00
Downloading cattrs-23.1.2-py3-none-any.whl (50 kB)
   ---------------------------------------- 50.8/50.8 kB 2.5 MB/s eta 0:00:00
Downloading joblib-1.3.2-py3-none-any.whl (302 kB)
   ---------------------------------------- 302.2/302.2 kB ? eta 0:00:00
Downloading inflect-7.0.0-py3-none-any.whl (34 kB)
Downloading exceptiongroup-1.1.3-py3-none-any.whl (14 kB)
Building wheels for collected packages: docopt
  Building wheel for docopt (setup.py): started
  Building wheel for docopt (setup.py): finished with status 'done'
  Created wheel for docopt: filename=docopt-0.6.2-py2.py3-none-any.whl size=13772 sha
256=e7591dff937a2a44bcd8e82296981b7447835be4674b3f66afcadeea0b727fb9
  Stored in directory: c:\users\steve\appdata\local\pip\cache\wheels\70\4a\46\1309fc8
53b8d395e60bafaf1b6df7845bdd82c95fd59dd8d2b
Successfully built docopt
Installing collected packages: sseclient-py, monotonic, events, docopt, url-normaliz
e, requests, rank-bm25, platformdirs, num2words, more-itertools, lazy-imports, jobli
b, exceptiongroup, canals, boilerpy3, backoff, tiktoken, scikit-learn, prompthub-py,
posthog, inflect, cattrs, requests-cache, quantulum3, accelerate, farm-haystack
  Attempting uninstall: requests
    Found existing installation: requests 2.27.1
    Uninstalling requests-2.27.1:
      Successfully uninstalled requests-2.27.1
  Attempting uninstall: joblib
    Found existing installation: joblib 1.1.0
    Uninstalling joblib-1.1.0:
      Successfully uninstalled joblib-1.1.0
  Attempting uninstall: scikit-learn
```

```
    Found existing installation: scikit-learn 1.0.2
    Uninstalling scikit-learn-1.0.2:
      Successfully uninstalled scikit-learn-1.0.2
Successfully installed accelerate-0.22.0 backoff-2.2.1 boilerpy3-1.0.6 canals-0.3.2 c
attrs-23.1.2 docopt-0.6.2 events-0.5 exceptiongroup-1.1.3 farm-haystack-1.19.0 inflec
t-7.0.0 joblib-1.3.2 lazy-imports-0.3.1 monotonic-1.6 more-itertools-10.1.0 num2words
-0.5.12 platformdirs-3.10.0 posthog-3.0.2 prompthub-py-4.0.0 quantulum3-0.9.0 rank-bm
25-0.2.2 requests-2.31.0 requests-cache-0.9.8 scikit-learn-1.3.0 sseclient-py-1.7.2 t
iktoken-0.4.0 url-normalize-1.4.3
```

In [61]:
```
#from haystack.reader.farm import FARMReader

#reader = TransformersReader(model_name_or_path="deepset/roberta-base-squad2",tokenize

model_name = "deepset/roberta-base-squad2"

# a) Get predictions
nlp = pipeline('question-answering', model=model_name, tokenizer=model_name)
QA_input = {
    'question': 'How many species of fish live in coral reefs?',
    'context': raw
}
res = nlp(QA_input)

# b) Load model & tokenizer
model = AutoModelForQuestionAnswering.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)

res
```

Out[61]:
```
{'score': 0.28468453884124756,
 'start': 3244,
 'end': 3257,
 'answer': '6,000â€“8,000'}
```

In [62]:
```
# a) Get predictions
QA_input = {
    'question': 'What is the most venomous fish?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[62]:
```
{'score': 0.612525463104248,
 'start': 24899,
 'end': 24913,
 'answer': 'reef stonefish'}
```

In [64]:
```
# a) Get predictions
QA_input = {
    'question': 'What sharks live in coral reefs?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[64]:
```
{'score': 0.1773730218410492,
 'start': 28108,
 'end': 28128,
 'answer': 'caribbean reef shark'}
```

In [65]:
```
# a) Get predictions
QA_input = {
    'question': 'What fish are poisonous?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[65]:
```
{'score': 0.5669053196907043,
 'start': 23682,
 'end': 23696,
 'answer': 'many reef fish'}
```

In [66]:
```
# a) Get predictions
QA_input = {
    'question': 'What fish can electrocute you?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[66]:
```
{'score': 0.010173969902098179,
 'start': 9045,
 'end': 9062,
 'answer': 'clown triggerfish'}
```

In [67]:
```
# a) Get predictions
QA_input = {
    'question': 'What species can sting you?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[67]:
```
{'score': 0.5866654515266418,
 'start': 27273,
 'end': 27286,
 'answer': 'the stargazer'}
```

In [68]:
```
# a) Get predictions
QA_input = {
    'question': 'What species are parasitic?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[68]:
```
{'score': 0.06776570528745651,
 'start': 17341,
 'end': 17353,
 'answer': 'damselfishes'}
```

In [69]:
```
# a) Get predictions
QA_input = {
    'question': 'What species are venomous?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[69]:
```
{'score': 0.3828498125076294,
 'start': 27273,
 'end': 27286,
 'answer': 'the stargazer'}
```

In [70]:
```
# a) Get predictions
QA_input = {
    'question': 'What species are known for attacking scuba divers?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[70]:
```
{'score': 0.005669948644936085,
 'start': 26369,
 'end': 26377,
 'answer': 'lionfish'}
```

In [71]:
```
# a) Get predictions
QA_input = {
    'question': 'What are common herbivorous fish?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[71]:
```
{'score': 0.33974504470825195,
 'start': 2116,
 'end': 2127,
 'answer': 'yellow tang'}
```

In [72]:
```
# a) Get predictions
QA_input = {
    'question': 'Why do fish camouflage themselves?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[72]:
```
{'score': 0.33112961053848267,
 'start': 8282,
 'end': 8309,
 'answer': 'lets them ambush their prey'}
```

In [73]:
```
# a) Get predictions
QA_input = {
    'question': 'How do fish get rid of their parasites?',
    'context': raw
```

```
}
res = nlp(QA_input)

res
```

Out[73]:
```
{'score': 0.2778742015361786,
 'start': 16126,
 'end': 16153,
 'answer': 'establish cleaning stations'}
```

In [74]:
```
# a) Get predictions
QA_input = {
    'question': 'What species have mutualistic relationships?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[74]:
```
{'score': 0.19288134574890137,
 'start': 21835,
 'end': 21861,
 'answer': 'sea anemones and clownfish'}
```

In [75]:
```
# a) Get predictions
QA_input = {
    'question': 'What species have commensalistic relationships?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[75]:
```
{'score': 0.1290857344865799,
 'start': 0,
 'end': 15,
 'answer': 'coral reef fish'}
```

In [76]:
```
# a) Get predictions
QA_input = {
    'question': 'Where in the world are coral reefs found?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[76]:
```
{'score': 0.11980749666690826,
 'start': 28031,
 'end': 28043,
 'answer': 'indo-pacific'}
```

In [77]:
```
# a) Get predictions
QA_input = {
    'question': 'What species engage in schooling?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[77]:
```
{'score': 0.235614612698555,
 'start': 27273,
 'end': 27286,
 'answer': 'the stargazer'}
```

In [78]:
```
# a) Get predictions
QA_input = {
    'question': 'What species are ambush predators?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[78]:
```
{'score': 0.5129007697105408,
 'start': 24899,
 'end': 24913,
 'answer': 'reef stonefish'}
```

In [79]:
```
# a) Get predictions
QA_input = {
    'question': 'Why are some coral reef fish colorful?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[79]:
```
{'score': 0.09240982681512833,
 'start': 81,
 'end': 145,
 'answer': 'coral reefs form complex ecosystems with tremendous biodiversity'}
```

In [80]:
```
# a) Get predictions
QA_input = {
    'question': 'Which species is blue?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[80]:
```
{'score': 0.4031549096107483,
 'start': 12029,
 'end': 12047,
 'answer': 'bluestripe snapper'}
```

In [81]:
```
# a) Get predictions
QA_input = {
    'question': 'What is the slowest species that lives in coral reefs?',
    'context': raw
}
res = nlp(QA_input)

res
```

Out[81]:
```
{'score': 0.6155030727386475,
 'start': 4939,
 'end': 4953,
 'answer': 'dwarf seahorse'}
```

In [34]:
```
for i in range(20):
    chatbot_models.append(5)

model_5_performance_list = ["Correct",
                            "Correct",
                            "Correct",
                            "Partially Correct",
                            "Incorrect",
                            "Partially Correct",
                            "Partially Correct",
                            "Correct",
                            "Correct",
                            "Correct",
                            "Correct",
                            "Correct",
                            "Correct",
                            "Incorrect",
                            "Correct",
                            "Incorrect",
                            "Correct",
                            "Incorrect",
                            "Partially Correct",
                            "Correct"]
```

6) Evaluation of Chatbot Performance Results

In [35]:
```
import pandas as pd


chatbot_performance_df = pd.DataFrame({'Correct': [100 * model_1_performance_list.cour
                                100 * model_2_performance_list.count("Correct")/len(moc
                                100 * model_3_performance_list.count("Correct")/len(moc
                                100 * model_4_performance_list.count("Correct")/len(moc
                                100 * model_5_performance_list.count("Correct")/len(moc
                     'Partially Correct': [100 * model_1_performance_list.count("Partia
                                100 * model_2_performance_list.count("Partially Correct
                                100 * model_3_performance_list.count("Partially Correct
                                100 * model_4_performance_list.count("Partially Correct
                                100 * model_5_performance_list.count("Partially Correct
                    'Incorrect': [100 * model_1_performance_list.count("Incorrect")/ler
                                100 * model_2_performance_list.count("Incorrect")/len(m
                                100 * model_3_performance_list.count("Incorrect")/len(m
                                100 * model_4_performance_list.count("Incorrect")/len(m
                                100 * model_5_performance_list.count("Incorrect")/len(m
                   index=['1', '2', '3', '4', '5'])
```

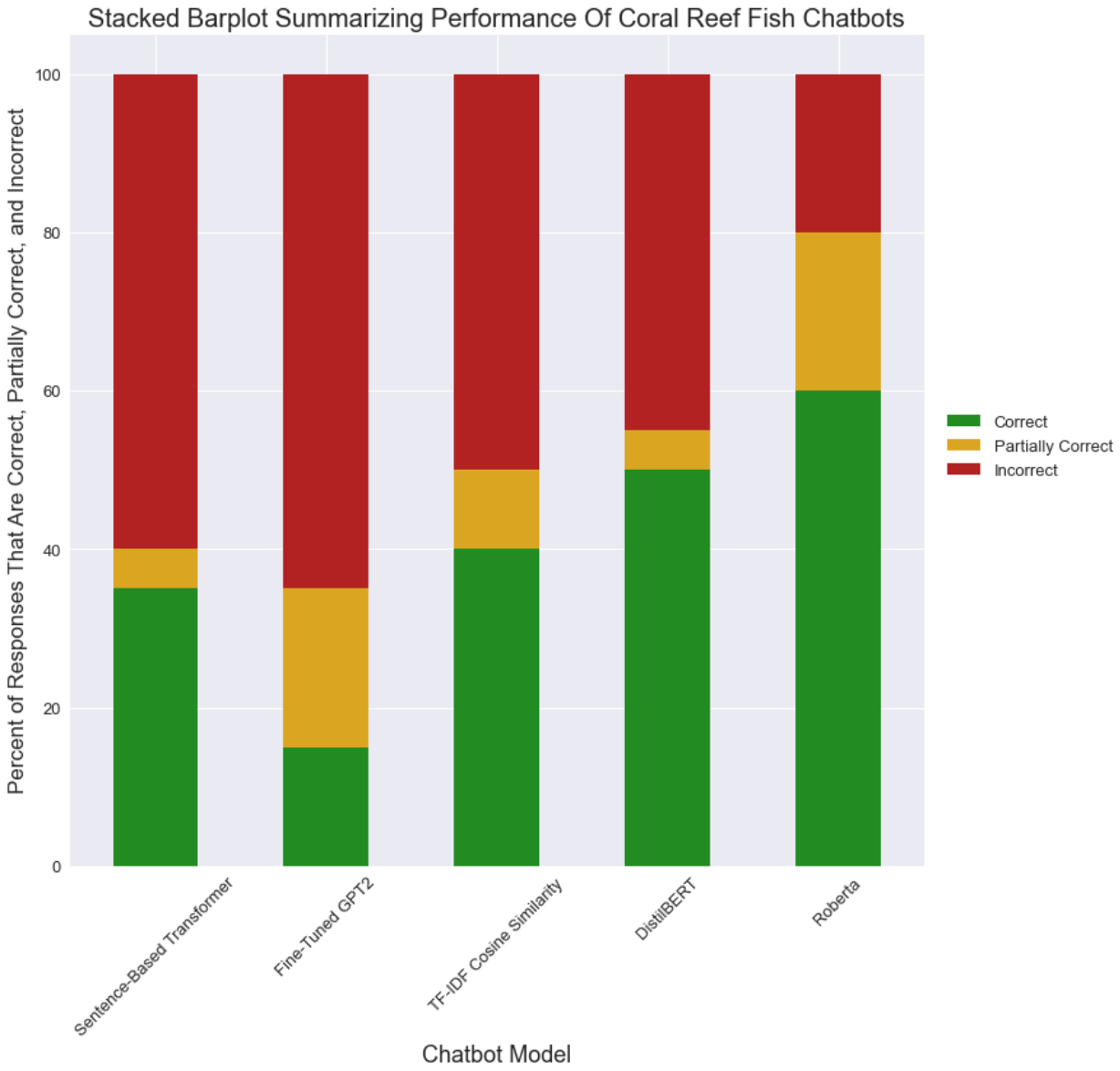
In [36]:
```
chatbot_performance_df
```

Out[36]:

	Correct	Partially Correct	Incorrect
1	35.0	5.0	60.0
2	15.0	20.0	65.0
3	40.0	10.0	50.0
4	50.0	5.0	45.0
5	60.0	20.0	20.0

In [40]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams

plt.style.use('seaborn')

ax = chatbot_performance_df.plot(kind = 'bar', stacked = True, figsize = (12, 12), for

ax.set_xticklabels(labels = ["Sentence-Based Transformer",
                             "Fine-Tuned GPT2",
                             "TF-IDF Cosine Similarity",
                             "DistilBERT",
                             "Roberta"], rotation = 45)
ax.set_xlabel("Chatbot Model", fontsize=18)
ax.set_ylabel("Percent of Responses That Are Correct, Partially Correct, and Incorrect
ax.set_title('Stacked Barplot Summarizing Performance Of Coral Reef Fish Chatbots', fo

plt.legend(bbox_to_anchor=(1.02, 0.55), loc='upper left', borderaxespad=0, fontsize =

plt.show();
```


Stacked Barplot Summarizing Performance Of Coral Reef Fish Chatbots
Percent of Responses That Are Correct, Partially Correct, and Incorrect
0
20
40
60
80
100
Sentence-Based Transformer
Fine-Tuned GPT2
TF-IDF Cosine Similarity
DistilBERT
Roberta
Chatbot Model
Correct
Partially Correct
Incorrect