(BLM3551-A) Yapay Zeka Proje Raporu

-- Ankü ChatBot --

18290059 Şuayip Deniz Talha Topay

18290620 Işınsu Çek

18290046 Altar Özcan

Bilgisayar Mühendisliği (Türkçe) 3. Sınıf

https://www.youtube.com/watch?v=AbSdNuS5TVE&feature=youtu.be

Projenin Amacı

Projemizin genel olarak amacı üniversite veya herhangi bir eğitim kurum veya kuruluşunda kullanılabilecek bir ChatBot oluşturmak. Bu ChatBot siteden bilgi almak isteyen birini sitenin karmaşık yapısında dolaşmaktan kurtaracak aynı zamanda kazandıracaktır. Ayrıca öğrendiği her yeni bilgi (kullanıcı girişleri) ile birlikte ortak problemi paylaşan başka kullanıcılara çözüm olacaktır.

Projeye Başlama ve Aşamalar

Projemize ilk olarak interaktif bir chatbot tasarlama amacı ile başladık. Sonraki adımda ise tasarlayacağımız bu chatbotun bildiklerine cevap vermesi ve öğrendikleri dahilinde yeni cevaplar oluşturmasına karar verdik. Sırada projemizi yazmak için uygun dile karar vermemiz gerekiyordu. Gerekli araştırmaları yaptıktan sonra Python dili ve Python kütüphaneleri/modülleri kullanarak projemizi gerçekleştirebileceğimiz kararına vardık. ChatterBot kütüphanesini kullanarak kullanıcı girişlerine otomatik cevap veren (bir rehber gibi davranan) bir ChatBot tasarlamaya başladık. Öncesinde ChatterBot'un nasıl işlediğini ve hangi karar mekanizmalarını kullandığını araştırdık.

ChatterBot

ChatterBot genel hatlarıyla, kullanıcı girişine uygun otomatik cevap veren chatbot yapmaya yarayan bir Python kütüphanesidir. ChatterBot çeşitli makine öğrenmesi algoritmaları kullanarak farklı cevaplar üretir. Bu programcıların otomatik cevap döndüren chat bot yapmalarını kolaylaştırır.

ChatterBot' un Dilden Bağımsız Olma Özelliği

ChatterBot'un dilden bağımsız olma özelliği herhangi bir dille eğitilip, öğrendiklerine göre cevap döndürmesine olanak sağlar. Bu kütüphane, makine öğrenmesine bağlı doğası sayesinde kullanıcılarla ve kaynaklarla ne kadar çok etkileşirse öğrendiklerine o kadar çok bilgi depolayacak daha sonra da amaca uygun cevap verecektir.

Biz projemizin kapsayıcı olup daha geniş bir kullanıcı kitlesine hitap etmesini istediğimiz için projemizi İngilizce girdi ve çıktılara göre geliştirdik. Programın İngilizce olması hem arka planda çalışan fonksiyonları açıklamamızda hem de ChatBot'umuzda girdi/çıktı uyumunun olmasını sağladı.

ChatterBot Nasıl Çalışır?

ChatterBot'un eğitilmemiş örneği, iletişim kurmaya dair hiçbir bilgiye sahip olmadan konuşmaya başlar. Kullanıcı her ifade girdiğinde, kütüphane girilen ifadeyi ve hangi girdiye karşılık olarak verilmiş olduğunu kaydeder. ChatterBot daha çok girdi aldıkça girilen ifadeye verebileceği cevapların sayısı ve verdiği cevabın doğruluğu/uygunluğu artmaktadır. Program ilk olarak, girdiye karşılık kendi veritabanında bildiği en yakın cevapları arar daha sonra bilinen yakın cevaplardan en uygununu seçerek bize karşılık verir.

ChatterBot'un işlem akış diyagramına bakacak olursak adımlar sırayla şu şeklide ilerlemektedir:

1) Girdi Alma Aşaması

   Herhangi bir kaynaktan girdi alınarak başlanır. Bu girdi konsol girdi, uygulama programlama veya bir başka tür bir veri seti olabilir.

2) Girdinin İşlenme Aşaması

   Girdi, mantık uyarlayıcılara (logic adaptors) girmeden önce bir önişlemci (preprocessors) aşamasından geçmektedir. Önişlemci, girdiyi değiştiren (üzerinde birtakım işlemler uygular) ardışık işlemlerdir. Bu işlemlere; ardışık boşlukları silme (whitespace karakterlerin silinmesi), girilen bazı ek/gereksiz karakterlerin yok sayılması ve özel karakterlerin ASCII karşılıklarına çevrilmeleri örnek olarak verilebilir. Önişlemci (preprocessor) aşamasında girilen dili algılamak gibi daha karmaşık işlemler de gerçekleştirilebilir. ChatterBot'ta bu önişlemcilere ek olarak kendi önişlemcimizi de tanımlayabiliriz. Preprocessor sadece birkaç gereksinime ihtiyaç duyan bir fonksiyondan ibarettir. Fonksiyon `ChatBot` örneği ve `Statement` örneği alarak `Statement` yine örneği döndürmelidir (return statement).

```python
def clean_whitespace(chatbot, statement):
    """
    Remove any consecutive whitespace characters from the statement text.
    """
    import re

    # Replace linebreaks and tabs with spaces
    statement.text = statement.text.replace('\n', ' ').replace('\r', ' ').replace('\t', ' ')

    # Remove any leeding or trailing whitespace
    statement.text = statement.text.strip()

    # Remove consecutive spaces
    statement.text = re.sub(' +', ' ', statement.text)

    return statement
```

#Ardışık boş karakterleri yok saymak için ChatterBot preprocessor örnek kodu

3) Mantık Uyarlayıcıları Aşaması

Mantık uyarlayıcıları (logic adapters) ChatterBot'un nasıl cevap seçtiğine karar veren mantık yapılarıdır. Bot'umuzda spesifik olarak kullanmak istediğimiz bir mantık uyarlayıcısı varsa, `logic_adapters` parametresine kullanmak istediğimiz mantık uyarlayıcısının erişim yolunu ayarlayarak/bağlayarak ulaşabiliriz.

```python
chatbot = ChatBot(
    "My ChatterBot",
    logic_adapters=[
        "chatterbot.logic.BestMatch"
    ]
)
```

#Örn: Bot'umuzda mantık uyarlayıcılarından "BestMatch" uyarlayıcısını kullanmak istersek yapmamız gereken işlem

Bot'umuza kullanması için birden fazla mantık uyarlayıcısı girmemiz mümkün. Böyle bir durumda bot, aralarından en çok güven değerine sahip (highest calculated confidence valuece) cevabı seçerek döndürecektir. Her birden fazla mantık uyarlayıcı aynı en yüksek değeri döndürürse, Bot'umuza ilk sırada yazılan uyarlayıcının cevabı diğer cevaplara kıyasla önceliğe sahip olup çıktı olarak verilecektir.

Bot'un bulduğu her cevabın bir güven değeri (confidence value) vardır. Herhangi bir cevabın güven değeri, alınan girdiye göre cevabın doğru olup olmadığı olasılığını belirtmektedir. Güven değeri en az 0 ve çok 1 olmak üzere bu aralıkta bir değer alır.

Kendimiz birden fazla mantık uyarlayıcı girmek yerine `MultiLogicAdapter` kullanarak ChatterBot'ta bulunan tüm mantık uyarlayıcıları arasından en iyi sonucu (en yüksek güven değerine sahip olan cevap) döndürebiliriz.

Confidence	Statement
0.2	Good morning
0.5	Good morning
0.7	Good night

Dikkat etmemiz gereken başka bir nokta ise birden fazla cevabın farklı güven değerleri ile döndürülmesi. Birden fazla mantık uyarlayıcısı aynı sonuca karar verdiğinde, bu cevap diğer cevaplara kıyasla öncelik kazanır.

Mantık Uyarlayıcıları Nasıl Cevap Seçer?

Mantık uyarlayıcılarının çalışma prensibi ChatterBot'un temelde nasıl çalıştığını belirlemektedir.

ChatterBot'ta bulunan çoğu mantık uyarlayıcıları girişe uygun çıktı oluştururken 2 temel adımı uygulamaktadırlar. İlk adım veritabanında girdi aranıp "Acaba bu girdiye karşılık daha önce belirlenen veya buna yakın olan bir girdi var mıydı?" kontrolü yapılır. Eşleşme yapıldıktan sonra ikinci adımda bilinen girdiye karşılık cevap aranır. Bilinen girdiye çoğunlukla birden çok cevap olacağından, ChatterBot uygun olanı seçmek için birkaç metot kullanır.

-- ChatterBot Yanıt Seçme Metotları --

Yanıt seçme metotları, mantık uyarlayıcıları tarafından birden fazla yanıt döndürüldüğü durumda uygun yanıtı seçmemizi sağlarlar.

```python
def get_most_frequent_response(input_statement, response_list):
    """
    :input_statement parametresi: chatbot'ta en yakın eşleşen girdi.
    :input_statement tipi: Statement

    :response_list parametresi: Aralarından seçim yapacağımız cevaplar listesi.
    :response_list tipi: list

    :return(çıktı): En çok karşılaşılma/tekrar edilme değerine sahip olan cevap.
    :return tipi: Statement
    """
    matching_response = None
    occurrence_count = -1

    logger = logging.getLogger(__name__)
    logger.info(u'Selecting response with greatest number of occurrences.')

    for statement in response_list:
        count = statement.get_response_count(input_statement)

        # Daha çok karşılaştığımız cevabı/cevapları sakla
        if count >= occurrence_count:
            matching_response = statement
            occurrence_count = count

    # En çok karşılaşılan ve eşleşen değeri seç
    return matching_response
```

Yanıtlar listesinde en çok tekrar eden yanıtı bize döndürür.

```python
def get_first_response(input_statement, response_list):
    """
    :input_statement parametresi: chatbot'ta en yakın eşleşen girdi.
    :input_statement tipi: Statement

    :response_list parametresi: Aralarından seçim yapacağımız cevaplar listesi.
    :response_list tipi: list

    :return(çıktı): Cevaplar listesindeki ilk durumu çıktı olarak veriyoruz.
    :return tipi: Statement
    """
    logger = logging.getLogger(__name__)
    logger.info(u'Selecting first response from list of {} options.'.format(
        len(response_list)
    ))
    return response_list[0]
```

Yanıtlar listesindeki ilk yanıtı bize döndürür.

```python
def get_random_response(input_statement, response_list):
    """
    :input_statement parametresi: chatbot'ta en yakın eşleşen girdi.
    :input_statement tipi: Statement

    :response_list parametresi: Aralarından seçim yapacağımız cevaplar listesi.
    :response_list tipi: list

    :return(çıktı): Cevaplar listesinde rastgele seçilen bir cevap.
    :return tipi: Statement
    """
    from random import choice
    logger = logging.getLogger(__name__)
    logger.info(u'Selecting a response from list of {} options.'.format(
        len(response_list)
    ))
    return choice(response_list)
```

Yanıtalar listesinden bize rastgele bir yanıt döndürür.

ChatterBot Hangi Makine Öğrenme Tekniklerini Kullanır?

ChatterBot cevap üretmek için birçok makine öğrenmesi tekniği kullanmaktadır. Spesifik algoritmalar kullanımı ChatBot'un ne sıklıkla kullanıldığına, ne amaca göre kullanıldığına ve hangi ayarlarda kullanıldığına bağlıdır. Genel olarak ChatterBot'un kaynak koduna baktığımızda makine öğrenmesi tekniklerine örnek olarak arama ve sınıflandırma algoritmaları ile karşılaşırız.

1) Arama Algoritmaları

   Arama yapma yapay zeka uygulamalarının en temel/ilkel formudur. Arama, ChatBot'un ne kadar hızlı ve verimli cevap vermesi aşamasında çok önemli bir rol oynamaktadır. ChatBot'un cevap bulmasına yardımcı olan bazı nitelikler şu şekildedir:

   - Girdinin elde hazır olan veriler ile olan benzerliği
   - Bilinen cevapların ne sıklıkla tekrar edildikleri
   - Girdinin, bilinen durumlar kategorisine girme olasılığı

ChatterBot buna benzer işlemleri yapmak için `statement_comparison_function` mantık uyarlayıcısını kullanmaktadır. Ancak istersek bu fonksiyonu kullanmak yerine ChatterBot'a başka algoritmalar uygulayabiliriz (implement). Karakter bazlı benzerlik aramak için `levenshtein_distance`, ifadelerde eşanlamlı benzerlik aramak için `synset_distance`, text girdisi içerisinde terim benzerliği bulmak için

`jaccard_similarity` metotlarını kullanabiliriz. Bu metotların arasından `levenshtein_distance` ifadeler arası en iyi eşleşmeyi bulmak için için kullanılır.

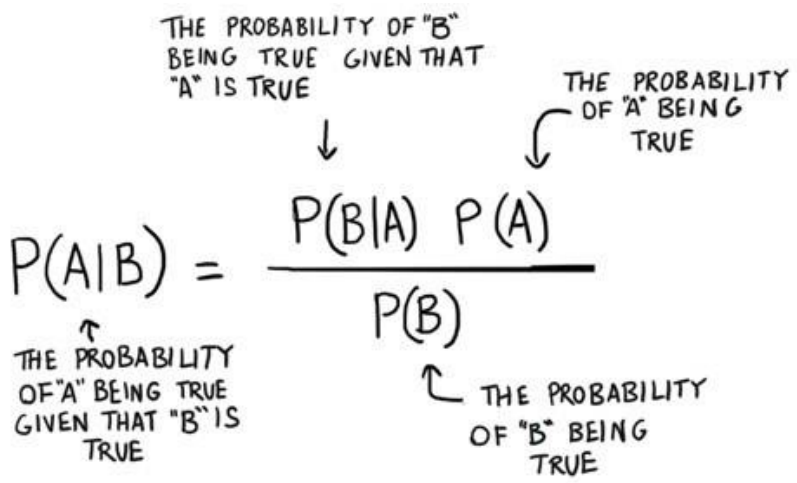
2) Sınıflandırma Algoritmaları

ChatterBot'ta bulunan birkaç mantık uyarlayıcısı giriş ifadesinin belirli kriterlere uyup uymadığının kontrolünü Naif Bayes Sınıflandırıcı (Naive Bayes Classifier) algoritmasına göre karar verir.

-- Bayes Teoremi –

Thomas Bayes tarafından bulunan koşullu olasılık hesaplama formülüdür.

Teoremin Tanımı: Bu teorem bir rassal değişken için olasılık dağılımı içinde koşullu olasılıklar ile marjinal olasılıklar arasındaki ilişkiyi gösterir.



P (A | B) = B olayı gerçekleştiğinde A olayının gerçekleşme olasılığı

P (A) = A olayının gerçekleşme olasılığı

P (B | A) = A olayı gerçekleştiğinde B olayının gerçekleşme olasılığı

P (B) = B olayının gerçekleşme olasılığı

-- Naif Bayes Sınıflandırıcı –

Naive Bayes sınıflandırıcısının temeli Bayes teoremine dayanır. Lazy (tembel) bir öğrenme algoritması olan Naif Bayes Sınıflandırıcısı aynı zamanda dengesiz veri kümelerinde de çalışabilir. Algoritma bir eleman için her durumun olasılığını hesaplar ve olasılık değeri en yüksek olana göre sınıflandırır. Naif Bayes Sınıflandırıcıları kısıtlı eğitim verileriyle çok başarılı işler çıkartabilirler. Test kümesindeki bir değerin eğitim kümesinde gözlemlenemeyen bir değeri varsa olasılık değeri olarak 0 verirler yani tahmin yapamazlar. Bu durum Zero Frequency (Sıfır Frekans) olarak bilinmektedir. Bu durumun üstesinden gelmek için düzeltme teknikleri (örneğin laplace düzeltme tahmini) kullanılabilir. Kullanım alanlarına örnek olarak gerçek zamanlı tahmin, öneri sistemleri ve duyarlılık analizi verilebilir.

AnküBot'un Detayları

```
bot = ChatBot(
    "Terminal",
    storage_adapter="chatterbot.storage.SQLStorageAdapter",
    logic_adapters=[

                "chatterbot.logic.MathematicalEvaluation",
                "chatterbot.logic.BestMatch"



    ],

    input_adapter="chatterbot.input.TerminalAdapter",
    output_adapter="chatterbot.output.TerminalAdapter",
    database="../database.db"

)
```

ChatBot sınıfının 'Terminal' adında bir bot isimli bir objesini oluşturduk.

Kullanacağımız mantık uyarlayıcılarına da burada karar verdik.

Öğrenilen girdi ve çıktıların tutulacağı bir veritabanı oluşturma işlemini de bu kısımda tanımladık.

```
trainer = ListTrainer(bot)



#trainer = ChatterBotCorpusTrainer(bot)


trainer.train([
    "Hi",
    "Hello!",
    "Hi there!",
    "Hello!",
    "Hello",
    "Hello!",
])

trainer.train([
    "How are you ?",
    "I am fine you ?",
])


trainer.train([
    "Thank you.",
    "No problem, how may I help you ?",
    "Thanks.",
    "No problem, how may I help you ?",
])

trainer.train([
    "Exam dates ?",
    "Dates are : 21.01.2021",
])
```

ChatterBot ChatBot objesinin eğitilmesi işlemini kolaylaştırmak için araçlar içermektedir. ChatterBot'u eğitme aşaması ChatBot'un veritabanına örnek diyalogları yüklemeyi içerir.

ChatBot eğiticisine veri kümeleri sağlandığında, bilgi grafında gerekli girdileri oluşturur. Böylelikle girdi ifadeleri ve cevaplar doğru bir şekilde gösterilir.

Training eğitim süreci için, ChatterBot'un kütüphanesinde tanımlı olan train() sınıfdan obje oluşturarak eğitme aşamasını gerçekleştirdik.

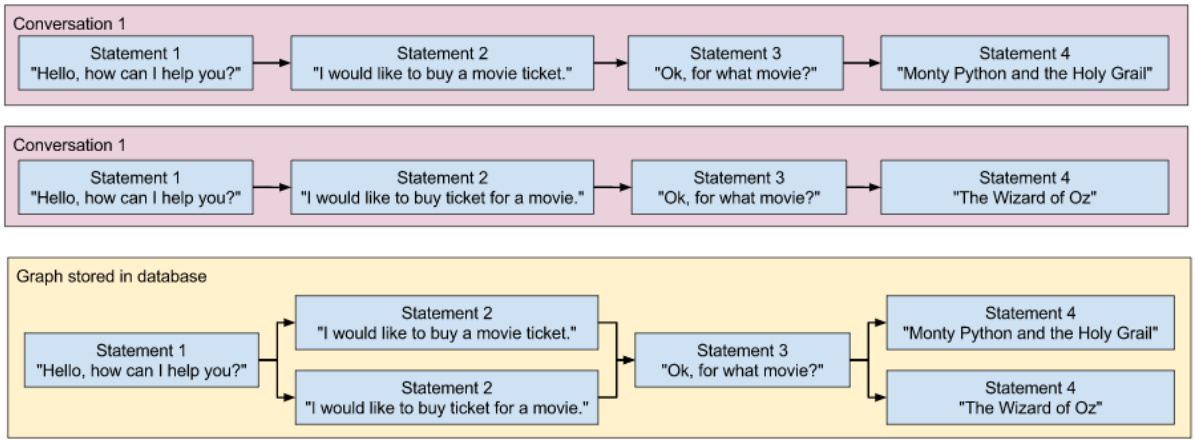
Bu süreçte olası konuşmaları temsil etmek adına kendi string listelerinden yararlandık.

```
trainer.train([
    'when are the exams?',
    'they start tomorrow',
    'do we have any exams coming up?',
    'tomorrow',
    'when are we having our exams?',
    'tomorrow',
    'when are the exams expected to begin?',
    'tomorrow'
])

trainer.train([
    "When does the finals week start?",
    "Finals week start at January 13, 2021.",
    "Thanks"
    "Can I help you with something else?",
])

trainer.train([
    "What are exam dates?",
    "Exam dates are : 21.01.2021",
    "Exam dates?",
    "Exam dates are : 21.01.2021",
    "What is the exam dates?",
    "Exam dates are : 21.01.2021",
    "Dates for the exams?",
    "Exam dates are : 21.01.2021",

])
```



Kaynak: https://chatterbot.readthedocs.io/en/stable/_images/training-graph.svg

```
while True:
        try:

                text=input('User: ')
                if(text=='quit' or text=='exit'):
                        break


                input_statement = Statement(text)
                response = bot.get_response(input_statement)




                if(response.confidence < 0.60):
                        print("\nThis answer has a low confidence level\n")
                        print('Confidence Number :',response.confidence)
                        print('\n Is "{}" a coherent response to "{}"? \n'.format(response.text,input_statement.text))


                        if get_feedback() is True:
                                print('Confidence level increased!')
                                bot.learn_response(response, input_statement)


                        else:
                                print('please input the correct one')
                                A=Statement(input('User: '))


                                #A.save()


                                bot.learn_response(A, input_statement)
                                print('Responses added to bot!')

                if(response.confidence > 0.60):
                        print("This answer has a high confidence level\n")
                        print('Confidence Number :',response.confidence)
                        print('\n',response)
```

Öncelikle kullanıcı ‘quit ’veya ‘exit ’yazdığında programın sonlanmasını sağlayan bir yapı oluşturduk.

Kullanıcının girdilerinin ChatterBot kütüphanesindeki fonksiyonlar tarafından işlenebilmesi için girdimizi `statement` tipine çevirdik.

Daha sonra yanıt seçme metodu sayesinde girdiye uygun AnküBot’tan cevap aldık.

Botumuzun çalışma prensibinden ötürü güven derecesi kontrolleri yapmaya karar verdik. Bu doğrultuda ChatBot’un güven seviyesini kontrol edip çıkan sonuçlara göre karar almasını sağlayacak bir yapı inşa ettik. Örneğin ChatBot, güven derecesi 0.60’tan düşük bir cevap vermeye karar verdiğinde kullanıcıya, vermiş olduğu cevabın uygun bir cevap olup olmadığını sorar. Eğer ChatBot’un verdiği cevap uygunsa, bu cevap güven derecesi artırılıp tekrar veri tabanına gönderilir. Aksi takdirde kullanıcıdan uygun cevabın girilmesi beklenir, yeni cevap veritabanına gönderilir.

```
User: hi there bot

This answer has a low confidence level

Confidence Number : 0

 Is "I am an AI ChatBot for Ankara University" a coherent response to "hi there bot"?

no
please input the correct one
User: hi there user
Responses added to bot!
```

Kaynakça:

- ChatterBot: https://chatterbot.readthedocs.io/en/stable/index.html

- ChatterBot GitHub: https://github.com/gunthercox/ChatterBot

- Vikipedi: https://tr.wikipedia.org/wiki/Naive_Bayes_sınıflandırıcısı

- Intelligent Online Tools : https://ai.intelligentonlinetools.com/ml/chatbots-examples-chatterbot/

- Tutorial Docs: https://www.tutorialdocs.com/tutorial/chatterbot/preprocessors.html