МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
ХЭРЭГЛЭЭНИЙ ШИНЖЛЭХ УХААН, ИНЖЕНЕРЧЛЭЛИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Анужингийн Сайнзолбоо

АЖИЛ ОЛГОГЧДЫН ӨГӨГДЛИЙН АНАЛИЗ СИСТЕМ ДЭЭР СУУРИЛСАН ЧАТ БОТ

(Chat bot based on sytem analysis of employers' data)

Мэдээллийн технологи (D061303)
Бакалаврын судалгааны ажил

Улаанбаатар

2022 оны 03 сар

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
ХЭРЭГЛЭЭНИЙ ШИНЖЛЭХ УХААН, ИНЖЕНЕРЧЛЭЛИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

АЖИЛ ОЛГОГЧДЫН ӨГӨГДЛИЙН АНАЛИЗ СИСТЕМ ДЭЭР СУУРИЛСАН ЧАТ БОТ

(Chat bot based on sytem analysis of employers' data)

Мэдээллийн технологи (D061303)
Бакалаврын судалгааны ажил

Удирдагч: _____ Др. Б.Хуягбаатар

Гүйцэтгэсэн: _____ А.Сайнзолбоо (18B1NUM1762)

Улаанбаатар

2022 оны 03 сар

Зохиогчийн баталгаа

Миний бие Анужингийн Сайнзолбоо ”АЖИЛ ОЛГОГЧДЫН ӨГӨГДЛИЙН АНАЛИЗ СИСТЕМ ДЭЭР СУУРИЛСАН ЧАТ БОТ” сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн зэрэг горилохоор дэвшүүлсэн болно.
- Энэ ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилохоор оруулж байгаагүй.
- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол ишлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг дипломын ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

ГАРЧИГ

ЗУРГИЙН ЖАГСААЛТ

ХҮСНЭГТИЙН ЖАГСААЛТ

Кодын жагсаалт

УДИРТГАЛ

Мэдээллийн технологи эрчимтэй хөгжиж буй өнөөгийн нийгэмд байгууллага үйл ажиллагаа явуулж эхэлсэн цагаасаа эхлэн өгөгдлийг үйлдвэрлэсээр байдаг. Тэдгээр өгөгдлийг байнга хадгалах нь өгөгдлийн сангийн нөөцөд хортой байдаг тул өгөгдөлд шинжилгээ хийх

1. СЭДВИЙН ТАНИЛЦУУЛГА

1.1 Оршил

Энэхүү бакалаврын судалгааны ажлын хүрээнд ”Ажил олгогчдын өгөгдлийн анализ систем дээр суурилсан чатбот” сэдвийн дагуу ажлын байрны мэдээллээр хангах Чатбот системийг хөгжүүлнэ. Ажлын байрны мэдээллийг Data Scraping аргын тусламжтайгаар, системд шаардлагатай мэдээллийг өгөгдлийн сангийн хэлбэрт оруулан бүтэцтэйгээр нэгтгэн авах бөгөөд үүнээс ажил горьлогчдын дунд байдаг түгээмэл асуултуудын хариултыг өгнө.
Мөн энэ системд машин сургалтын арга болох Language Understanding-ийг ашиглан хэрэглэгчийн асуултыг таамаглаж оновчтой хариулт өгөх боломжийг олгох юм.

1.2 Зорилго

Ажлын горьлогчдын хэрэгцээт асуултад хариулж, ажлын байрны хүртээмжийг нэмэгдүүлэхэд энэхүү системийн гол зорилго оршино.

1.3 Зорилт

Дээрх зорилгод хүрэхийн тулд дараах зорилтуудыг тавьсан. Үүнд:

- Ашиглагдах технологиудыг сонгох, судлах
- Ижил төстэй системийн судалгаа хийх
- Системийн шинжилгээ хийх
- Системийг зохиомжлох
- Системийг хөгжүүлэх, сайжруулалт хийх

1.4 Алсын хараа

Ажлын байрны дэлгэрэнгүй мэдээллийг цуглуулснаар цаашид тэдгээрт шинжилгээ хийж эрэлттэй ажлын байр, өндөр цалинтай ажлын байр гэх зэрэг мэдээллүүдийг систем хэрэглэгчдэд хүргэх боломжтой юм.

2. СИСТЕМИЙН СУДАЛГАА

2.1 Системийн судалгаа

Сонгосон сэдэв болох ”Ажил өгөгдлийн анализ систем дээр суурилсан чат бот”-ын сэдвийн судалгааг хийхдээ эхлээд бараа материалын менежментийн системийн тухай болон ERP систем дээрх Бараа материалын модулийн үйл ажиллагааг судалсан. Дараагийн хэсэгт ижил төстэй програмуудын судалгааг, ашиглах технологийн судалгааг хийсэн болно.

2.1.1 Чатбот систем

2.2 Ижил төстэй системүүд

2.3 Технологийн судалгаа

3. СИСТЕМИЙН ШИНЖИЛГЭЭ

3.1 Бизнесийн үйл ажиллагааны шинжилгээ

3.2 Хэрэглэгч

3.3 Функционал шаардлага

3.4 Функционал бус шаардлага

3.5 Use case диаграм

4. СИСТЕМИЙН ЗОХИОМЖ

4.1 Өгөгдлийн сангийн диаграм

4.2 Өгөгдлийн элемент

5. ХЭРЭГЖҮҮЛЭЛТ, ҮР ДҮН

5.1 Хөгжүүлсэн байдал

Bibliography

А. ҮЕЧИЛСЭН ТӨЛӨВЛӨГӨӨ

Батлав.

МКУТ-ийн эрхлэгч:................................./док. проф. Н.Оюун-Эрдэнэ/ 2022 оны 02 сарын 11

Монгол нэр Ажил олгогчдын өгөгдлийн анализ
систем дээр суурилсан чат бот
Англи нэр Chat bot based on system analysis
of employers' data
Сэдэвт бакалаврын судалгааны ажлын
7 хоногийн үечилсэн төлөвлөгөө

Хугацаа: **2022.02.07**-оос **2022.05.06** хүртэл **13** долоо хоног

№	Хийх ажил	Долоо хоног	1	2	3	4 Явц I	5	6	7	8 Явц II	9	10	11	12 Урьдчилсан хамгаалалт	13	14 Жинхэнэ хамгаалалт	Тайлбар
1	**Онолын судалгаа**	Scrapper tool															
		Bot tool															
2	**Өгөгдөл цуглуулалт**	Цуглуулах код бичих															
		Өгөгдлийг баазруу цуглуулах															
3	**Системийн шинжилгээ**	Системийн шаардлага тодорхойлох															
		Хэрэглэгчийн шаардлага тодорхойлох															
5	**Системийн зохиомж**	Өгөгдлийн сангийн зохиомж															
6	**Хэрэгжүүлэлт**	Чат бот хөгжүүлэлт															
		Өгөгдөлд анализ хийх сайжруулах															
7	**Бичиг баримт**	Тайлан боловсруулах															

Тайлбар: Төслийг хэрэгжүүлэх төлөвлөгөөг 7 хоногийн давтамжтайгаар хийж тод хараар будаж тэмдэглэнэ. Хийх ажил дэд хэсэгтэй байвал уг ажилд зарцуулах хугацааг хуваан төлөвлөж болно. Ажлын эхлэх төгсөх хугацаа хоорондоо давхцаж болно. Ажлын гүйцэтгэлийг дүгнэж тэмдэглэгээ хийх боломжтой байхаар "7 хоног" баганыг үүсгэнэ.

Зөвшөөрсөн: Удирдагч багш ..../Б. Хуягбаатар/

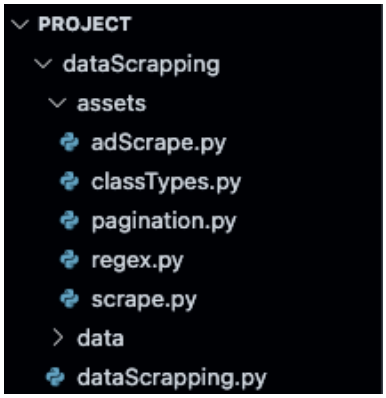
Боловсруулсан: Оюутан .... /Мэдээллийн технологи А. Сайнзолбоо/
Оюутны ID: 18b1num1762
Холбогдох утас: 91990388

Зураг A.1: Бакалаврын судалгааны ажлын үечилсэн төлөвлөгөө

В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

В.1 Өгөгдөл цуглуулалт

Өгөгдөл цуглуулах програм нь дараах бүтэцтэй байх бөгөөд assets доторх кодууд нь үндсэн кодыг ажлуулахад туслах функцууд байна.



Зураг В.1: Фолдерийн бүтэц

В.1.1 Үндсэн өгөгдлийг цуглуулах эх код

```
from datetime import date
import time
from assets.classTypes import Category
from assets.scrape import UseBeautifulSoup as useScrape
from assets.adScrape import advertisementScrape as useAdScrape
from assets.pagination import createLinkList as createLinkList

start_time = time.time()
initialUrl = 'https://www.zangia.mn/'
today = str(date.today())
# all categories set
categorySet = set()
# all advertisement's link set
adUrlDict = {}
# all ads object set
adsSet = set()

# scrape initial links
soup = useScrape(initialUrl)
navigatorList = soup.find_all('div', class_='filter')
for navigator in navigatorList:
    if navigator.find('h3').text.strip() != '   ,    ':
        continue
    # ALL CATEGORY LINKS
    categoryList = navigator.find_all('div')
```

```
26
27 for categoryItem in categoryList:
28     categories = categoryItem.find('a')
29     url = initialUrl + categories['href']
30     tempCategory = Category(url, categories.text, '')
31     print('CATEGORY LINK SCRAPED! ', url)
32     soup = useScrape(url)
33     subCategory = soup.find('div', class_='pros')
34     # ALL SUBCATEGORY LINKS
35     subCategoryList = subCategory.find_all('a')
36     for subCategoryItem in subCategoryList:
37         subCategoryUrl = initialUrl + subCategoryItem['href']
38         tempSubCategory = Category(
39             subCategoryUrl, subCategoryItem.text, tempCategory.name)
40         categorySet.add(tempSubCategory)
41
42 for categoryItem in categorySet:
43     if categoryItem.parentId == '':
44         continue
45     soup = useScrape(categoryItem.url)
46     hasPagination = soup.find('div', class_='page-link')
47     pagesUrl = []
48     if hasPagination != None:
49         pagesUrl = createLinkList(hasPagination, categoryItem.url)
50     else:
51         pagesUrl.append(categoryItem.url)
52     for pageUrl in pagesUrl:
53         soup = useScrape(pageUrl)
54         ads = soup.find_all('div', class_='ad')
55         # CREATE UNIQUE AD DICTIONARY
56         for ad in ads:
57             adUrl = initialUrl+ad.find('a', class_=None)['href']
58             adUrlDict[adUrl] = categoryItem
59     print(pagesUrl)
60     pagesUrl.clear()
61
62 file = open(today+'adScrape.csv', 'w', encoding='utf-8')
63 file.write('Parent Category Name' + '\t' +
64            'Category Name ' + '\t' +
65            'Link' + '\t' +
66            'Employee Company' + '\t' +
67            'Title' + '\t' +
68            'Roles' + '\t' +
69            'Requirements' + '\t' +
70            'Additional Info' + '\t' +
71            'City/Province' + '\t' +
72            'District' + '\t'
73            'Level' + '\t' +
74            'Type' + '\t' +
75            'Min Salary' + '\t' +
76            'Max Salary' + '\t' +
77            'Is Dealable' + '\t' +
```

```
            'Address' + '\t' +
            'Phone' + '\t' +
            'Fax' + '\t' +
            'Ad Added Date' + '\n')
print(adUrlDict)
for adUrl in adUrlDict:
    print(adUrl)
    try:
        tempAdItem = useAdScrape(adUrl)
        tempAdItem.setCategory(adUrlDict[adUrl])
        file.write(
            tempAdItem.category.parentId+'\t' +
            tempAdItem.category.name+'\t' +
            tempAdItem.url+'\t' +
            tempAdItem.company+'\t' +
            tempAdItem.title+'\t' +
            tempAdItem.roles+'\t' +
            tempAdItem.requirements+'\t' +
            tempAdItem.additionalInfo+'\t' +
            tempAdItem.city+'\t' +
            tempAdItem.district+'\t' +
            tempAdItem.level+'\t' +
            tempAdItem.type+'\t' +
            tempAdItem.minSalary+'\t' +
            tempAdItem.maxSalary+'\t' +
            tempAdItem.isDealable+'\t' +
            tempAdItem.address+'\t' +
            tempAdItem.phoneNumber+'\t' +
            tempAdItem.fax+'\t' +
            tempAdItem.adAddedDate+'\n')
        del tempAdItem
    except:
        print('Ad writing error')
file.close()
print("--- %s seconds ---" % (time.time() - start_time))
```

Код B.1: Бүх өгөгдлийг цуглуулах - dataScrapping.py

B.1.2 Нэг зарын шаардлагатай бүх мэдээллийг цуглуулах код

```
import re
from .classTypes import Advertisement
from .scrape import UseBeautifulSoup as useScrape


def listScraper(sections, key) -> str:
    content = []
    for section in sections:
        subTitle = section.find('h2', class_=None).text
        if key != subTitle:
            continue
        div = section.find('div', class_=None)
```

```
        children = div.next_element

        while(children != None):
            try:
                content.append(textStrip(children.text))
                children = children.next_sibling
                continue
            except:
                print('An error occured')
            children = children.next_sibling
        content = [s for s in filter(listFunc, content)]
    if not content:
        return ''
    return ' '.join(content)


def textStrip(text) -> str:
    pattern = re.compile('[\r\n\xa0\t ]+', re.MULTILINE | re.IGNORECASE
       )
    return pattern.sub(' ', text.strip())


def listFunc(e):
    return len(e) != 0


def singleItemScraper(sections, key, subKey) -> str:
    for section in sections:
        subTitle = section.find('h2', class_=None).text
        if key != subTitle:
            continue
        div = section.find_all('div', class_=None)
        for item in div:
            if item.next_element.text == subKey:
                return textStrip(item.find('span').text)
    return 'None'


def salaryScraper(salary):
    isDealable = ''
    k = re.split(r'[^\d,]+', salary, 2, re.IGNORECASE)
    if len(k) < 2:
        [a] = k[0:1]
        return a, a
    [a, b] = k[0:2]
    if len(k) > 2:
        isDealable = '     '
    return a, b, isDealable


def locationScrapper(location):
    city = ''
```

```
    district = ''
    k = location.split(',')
    if len(k) < 2:
        city = k[0]
        return city, district
    [city, district] = k[0:2]
    return city, district


def advertisementScrape(url) -> Advertisement:
    soup = useScrape(url)
    advertisement = Advertisement(url, soup.find('h3').text.strip())
    companyTitle = soup.find('div', class_='nlp').find('td')
    for item in companyTitle:
        try:
            if item.name == None:
                advertisement.company = textStrip(item.text)
        except:
            print('Company name scrape error')
    # advertisement.company = textStrip(company)

    # all items
    sections = soup.find_all('div', class_='section')
    advertisement.roles = listScraper(
        sections, '            ')
    advertisement.requirements = listScraper(
        sections, '                ')
    advertisement.additionalInfo = listScraper(
        sections, '        ')
    advertisement.level = singleItemScraper(sections, '   ', '   ')
    advertisement.type = singleItemScraper(sections, '   ', '   ')
    minSalary, maxSalary, isDealable = salaryScraper(
        singleItemScraper(sections, '   ', '   '))
    city, district = locationScrapper(
        singleItemScraper(sections, '   ', '    '))
    advertisement.minSalary = minSalary
    advertisement.maxSalary = maxSalary
    advertisement.isDealable = isDealable
    advertisement.city = city
    advertisement.district = district
    advertisement.address = singleItemScraper(sections, '      ', '  ')
    advertisement.phoneNumber = singleItemScraper(
        sections, '      ', '  ')
    advertisement.fax = singleItemScraper(
        sections, '      ', '  ')
    advertisement.adAddedDate = singleItemScraper(
        sections, '       ', '          ')
    print('SINGLE AD SCRAPPING DONE!!!', url)

    return advertisement
```

Код B.2: Нэг зарын өгөгдлийг цуглуулах - adScrape.py

B.1.3 Цуглуулах өгөгдлийн төрөл

```
class Category:
    url = ''
    name = ''
    parentId = ''

    def __init__(self, url, name, parentId='None') -> None:
        self.url = url
        self.name = name
        self.parentId = parentId

    def getUrl(self) -> str:
        return self.url


class Advertisement:
    category = Category
    url = ''
    company = ''
    title = ''
    # ListInfo
    roles = ''
    requirements = ''
    additionalInfo = ''
    # OtherInfo
    city = ''
    district = ''
    level = ''
    type = ''
    minSalary = ''
    maxSalary = ''
    isDealable = ''
    # ContactInfo
    address = ''
    phoneNumber = ''
    fax = ''
    adAddedDate = ''

    def __init__(self, url, title) -> None:
        self.url = url
        self.title = title

    def setCategory(self, category) -> None:
        self.category = category
```

Код B.3: Өгөгдлийн төрөл - classTypes.py

B.1.4 Url хаягийн html-ийг авах функц

```
class Category:
    url = ''
```

```
    name = ''
    parentId = ''

    def __init__(self, url, name, parentId='None') -> None:
        self.url = url
        self.name = name
        self.parentId = parentId

    def getUrl(self) -> str:
        return self.url


class Advertisement:
    category = Category
    url = ''
    company = ''
    title = ''
    # ListInfo
    roles = ''
    requirements = ''
    additionalInfo = ''
    # OtherInfo
    city = ''
    district = ''
    level = ''
    type = ''
    minSalary = ''
    maxSalary = ''
    isDealable = ''
    # ContactInfo
    address = ''
    phoneNumber = ''
    fax = ''
    adAddedDate = ''

    def __init__(self, url, title) -> None:
        self.url = url
        self.title = title

    def setCategory(self, category) -> None:
        self.category = category
```

Код B.4: Scrape хийх функц - classTypes.py