Литература

1. Проблема молодежной безработицы в Казахстане [Электронный ресурс] http://www.ipr.kz/analytics/1/1/216

1. *Миронов М.В.,* Кондратьев Э.В. Опыт создания информационной системы для обеспечения трудоустройства выпускников высших учебных заведений // Управление инвестиционными процессами в экономике. Пенза, 2000. - С. 202-203.

2. *Мацяшек А.,* Анализ требований и проектирование систем. Разработка информационных систем. - М: Вильямс, 2002 - 432 с.

3. *Орлов С.А.,* Проектирование информационных систем, 2003.-465с.

УДК: 004.420

ҚАЗАҚ ТІЛІНДЕГІ МӘТІНДІ ТҮЗЕТУ

Үшкемпір Арайлым

Сулейман Демирель атындағы университет

araisdu@gmail.com

Түйін

Бұл жұмыс қазақша мәтіндегі орфографиялық қателерді түзетеді. Нақтырақ айтатын болсақ, мәтін енгізілген кезде түрлі қателер кетуі мүмкін. Жұмыстың мақсаты – сол қателерді дұрыстау жолдарын қарастырып , мүмкін болатын нұсқаларды ұсыну.

Кіріспе

Мәтінді енгізгенде екі түрлі қате кетуі мүмкін: біріншісі жазу тәртібін білмей қате жіберу, екіншісі енгізудегі ұқыпсыздық: сөзде әріп қалдырып кету (Мысалы: анықама) не артық жазу (Мысалы: апнықтама) сонымен қатар іргелес тұрған пернені басып жіберуден кететін қате (көбінде смартфондарда кездеседі). Бұл қателерді дұрыстау Google search және Microsoft word бағдарламаларында кіршіксіз жұмыс істейді. Алайда, бұл өте керекті амал тек қана ағылшын және басқа да тілдер үшін жұмыс істейді; қазақ тілі еш қарастырылмаған. Оған қоса қазіргі уақытта шығарылып жатқан қазақша кітаптарды адамдар өздері тексереді. Тексеру барысында өте көп уақыт кетеді. Бұлай тексерген күннің өзінде кітаптардан көптеген қателер кездесіп жатады

Төмендегі көрсетілген тәсіл енгізілген мәтіннен екі символдың арасында орналасқан сөзді тауып береді. Бұл тәсілдің жүзеге асуы өте қарапайым, бірақ аса сенімді емес. Тәсіл (http://norvig.com/spellcorrect.html) сайтында Питер Норвиг тарапынан толық сипатталған және Python тілінде жүзеге асырылған. Сайттағы жұмыс СДУ оқытушысы Ардақ Шалқарбайұлы тарапынан орыс тілінде сұрыпталып беріледі, және орындалуы java тіліне ауыстырылған. Бұл жұмыста жоғарыда көрсетілген адамдардың еңбектерін негізге алдық.

Жүзеге асыру

Бұл жобаны жүзеге асыру үшін *CorrectingMachine* класын құрамыз. Бұл класс екі public *train* және *correct* тәсілінен тұрады. Бірінші тәсілде тексерілетін енгізілген мәтінді бастапқы мәтіннен қарайды. Екінші тәсіл қате енгізілген мәтінді түзету үшін бастапқы мәтіннен тікелей үміткерлер іздейді.

Қарапайым мысал:

```
CorrectingMachine cm=new CorrectingMachine();cm.train(“Туған ауыл, сен үшін тер төкпедім, Кешір мені, кеш мені -сен Текті елім! Бір сағыныш мазалап саған деген, Жүреді ылғи жанымды өртеп менің. Туған ауыл. Бәрі есте... Балалық шақ, Қалаға кеп қаңғырдық қалам ұстап. Кешір мені күнәсіз балалығым, Қала алмадым ауылда саған ұқсап.”);
```

```
ArrayList<String> corrections = cm.correct(“ен”);
System.out.println(corrections);
```

Мысалдағы дұрыс жазылмаған «ен» сөзін алдындағы бес сөйлемді негізге ала отырып үміткерлер іздейді. Егер енгізген сөз базада жоқ болса онда программа үміткерлер шығармайды. Мысалы қазіргі жағдайда «мен» сөзіннен «м» әріпін жіберуден қате жіберген болсақ, онда бізге программа мынандай «мен» , «сен» үміткерлерін қайтарады.

Бірінші бөлім: мәтінді өңдеу

Біздің программамыз базадан қолданылған сөздерді алу үшін, сөйлемді сөздерге бөлеміз тек бұған сандар мен символдар кірмейді. Бұл үшін мәтінді тыныс белгілеріне (яғни бос орын, үтір, нүкте) қарап бөлетін *StringTokenizer* класын қолданамыз. Сандар мен символдардан тұранын сөзді шығарып тастау үшін *Regular Expressions*-ды қолданамыз. Программада Boolean мән қайтаратын *aripter()* тәсілі бар. Бұл тәсіл сөздің тек әріптен тұратынын не тұрмайтыннын тексереді, егер aripter() true мәнін қайтарса бұл сөзді кездесетін сөздер қатарына қосады.

```
 private void parseWords(String text) {
        text = text.toLowerCase();
        StringTokenizer stt = new StringTokenizer(text);
        while (stt.hasMoreTokens()) {
            String word = stt.nextToken();
            char ch;
            boolean bar=false;
            for (int i = 0; i < word.length(); i++) {
                ch = word.charAt(i);
                if (bar==aripter(ch)) break;
            }
            words.add(word);
}
              }
```

Келесі қадам: әр сөздің неше рет кездесетінін санау. Бұл үшін кіліт ретінде сөзді ал мәні ретінде сөздің қанша рет кездесетін санды сақтайтын *HashMap* құрамыз

```
   for (int i=0;i<words.size();i++){
if (!nwords.containsKey(words.get(i))){
int s = Collections.frequency(words, words.get(i));
nwords.put(words.get(i), s);
}
}
```

Осымен бірінші бөлім аяқталды, енді үлкен бір кітапты программамызға оқуға беруге болады.

Екінші бөлім: мәтінді тексеру

Біріншіден бұл қатар базада бар ма жоқ па соны тексереміз

```
 if (words.contains(sword)){
candidates.add(sword);
return candidates;
}
```

Келесі кезеңде біз енгізілгеннен бір символға айырмашылығы бар қатарды табуымыз керек. Енгізілген мәтіннен бір символдың қалып кетуінен туған қатарды табамыз.(*жету* - *жту*)

```
for (int i=0;i<a.length();i++){
```

```
String del = a.substring(0,i)+a.substring(i+1);
deletes.add(del);
        }
```

Қатар келген символдардың орын ауысуынан туған қатарлар (*жарқырау- жарқрыау*)

```
ArrayList<String> transposes = new ArrayList<String>();
for (int i=0;i<a.length()-2;i++){
String transpose =
a.substring(0,i)+(char)a.charAt(i+1)+(char)a.charAt(i)+a.substring(i+2);
transposes.add(transpose);
}
```

Әріптердің ауысып кетіп пайда болған қатар (Мысалы: *және-жіне*)

```
ArrayList<String> replaces = new ArrayList<String>();
for (int i=0;i<a.length();i++){
for (int j=0;j<26;j++){
String replace = .substring(0,i)+(char)('a'+j)+a.substring(i+1);
replaces.add(replace);
}
  }
```

Артық әріптің кіріп кету арқылы құлылған қатар(Мысалы: *мен-маен*)

```
ArrayList<String> inserts = new ArrayList<String>();
    for (int i=0;i<a.length();i++){
    for (int j=0;j<26;j++){
    String insert = a.substring(0,i)+(char)('a'+j)+a.substring(i);
    inserts.add(insert);
    }
```

Мәтінді теріп жатқанда іргелес тұрған пернені басу арқылы пайда болған қате. Бұл тәсілде біз тек әріптің екі шетіндегі әріптерді ғана алдық (Мысалы: *маған-иағаи*)

```
       ArrayList<String> irgeles = new ArrayList<String>();
            for (int i = 0; i < a.length(); i++) {
                for (int j = 0; j < 41; j++) {
                    String insert = "";
                    String insert2 = "";
                    if (a.charAt(i) == aripMas[j][1]) {
        insert = a.substring(0, i) + "" + aripMas[j][0] + "" + a.substring(i + 1,
a.length());
                        irgeles.add(insert);
      insert2 = a.substring(0, i) + "" + aripMas[j][2] + "" + a.substring(i + 1,
a.length());
                        irgeles.add(insert2);
                    }
                }
            }
```

Осылайша алынған сөздерді қайталамай бір тізімге жинастырамыз. Ол үшін *HashSet* қолданылады

```
            HashSet<String> candidates =new HashSet<String>();
      candidates.addAll(deletes);
      candidates.addAll(transposes);                          candidates.addAll(replaces);
                                            candidates.addAll(inserts)
                                                      return candidates;
```

Қолданылған сөздерден іздейміз және мәтінде кездескеніне қарап *sorting* жасаймыз.

```
HashSet<String>  c = edit1(sword);
```

```
    if (knownWords(c).size()>0){
    candidates.addAll(knownWords(c));
    Collections.sort(candidates,new NWordsComparator());
    return candidates;
    }
```

Егер берілген қатарлардан белгіленген сөзді таппасақ, алдынғы тәсіл арқылы қатардың табу тәсілімен өзгертеміз

```
HashSet<String> x=new HashSet<String>();
    for (String i : c){
    x.addAll(edit1(i));
      }
```

Және базада бар қатарды іздейміз

```
HashSet<String> z = knownWords(x);                                                    if
(z.size() > 0) {
              candidates.addAll(z);
    Collections.sort(candidates, new CorrectingMachine.NWordsComparator());
return candidates;
      }
```

Егер бұдан да таппасақ бос қайтарамыз.

Қортынды

Бұл тәсіл өте қарапайым, бірақ бірнеше кемшіліктері бар. Бірінші кемшілігі: ол тек NWE (non-word errors) ғана табуға көмектеседі, ал RWE (real word errors) ешқалай да таба алмайды. Яғни сіз “like sports”-тың орнына “like spots” деп жазсаныз, онда бұл жағдайда программа көмек бере алмайды, себебі “spots” сөзі де базада кездеседі. Программа сөздікте көп кездескен сөзді алғандықтан смартфондарға мүлдем келмейді, өйткені смартфондарда қате көбнесе іргелес тұрған пернелерден кетеді. Бұл жұмыста екінші кемшілік қарастырылып, шешу жолын ұсынған. Жоғарыда айтылған алғашқы жағдайды алдағы уақытта шешімін табуға тырысамыз. Сонымен қатар негізгі мақсатымыз - android платформасында қазақ тілінде арнайы пернетақта шығарып соның ішінде осы жұмысты пайдалану.

УДК: 004.420

М.Уатбаев, магистрант
Международный университет информационных технологий
Алматы, musa.uatbayev@gmail.com

Средства управления Web-сервисами

1. Информационная безопасность

Безопасность Web-приложений уже не первый год является важной составляющей в защите информационных систем. В настоящее время мы являемся свидетелями нового витка развития Web-технологий. Сейчас наблюдается тенденция активного переноса стандартных клиент-серверных приложений в Web-среду. Более того, в Web-среду переносятся такие разработки, как графические редакторы и офисные приложения, к тому же, работать с подобными системами становится доступным одновременно более чем одному человеку. Примерами могут послужить многочисленные приложения компании «Google», доступные средствами стандартных браузеров и не требующих установку дополнительного программного обеспечения, система электронного правительства, например e-gov.kz, которая активно развивается в РК. Ранее мы и подумать не могли, что подобные технологии так быстро начнут обширно применяться в нашей повседневности. Web-технологии заслужено отстояли право произвести революцию в мире