Lemmatiserare för okända ord

Boel MATTSSON
Lunds Tekniska Högskola
Lund, Sverige
d03bm@efd.lth.se

Abstract

En lemmatiserare hittar grundformen för ett böjt ord. Detta projekt har går ut på att göra en lemmatiserare för okända ord. En klassificerare har använts för att lösa problemet.
Rapporten beskriver bakgrunden till problemet och ger en kort beskrivning av hur en klassificerare fungerar. Träningsmängder och testmängder till klassificeraren har tagits från SUC-korpusen. Träningsmängder med olika antal ord har använts.
Resultaten visar att fler ord i träningsmängden ger bättre resultat och att denna metod fungerar bra.

1 Introduktion

Ibland stöter man på ord som inte påträffats tidigare. I det svenska språket kan man bilda många nya ord genom t ex sammansättingar. Orden *kräm* och *bluffen* kan sättas samman till ordet *krämbluffen*. Att hitta lemmat (grundformen) till ordet *bluffen* är enkelt eftersom *bluffen* är ett känt ord som finns i flera korpus. Detta gäller däremot inte för *krämbluffen*.

Syftet med projektet är att göra ett program som gissar lemmat för okända ord. Jag har begränsat mig till de öppna ordklasserna - substantiv, verb och adjektiv – eftersom det är i dessa ordklasser nybildning av ord oftast sker.

Metoden för att lösa lemmatiseringsproblemet har varit att formulera det som ett klassificeringsproblem. Det innebär att ett ord som *krämbluffen* betraktas som ett ord som tillhör den klass där man ska ta bort ändelsen *”-en”* för att få lemmat *krämbluff*. Syftet har varit att undersöka om detta angreppssätt är lämpligt.

Denna rapport är upplagd på följande vis. Avsnitt två beskriver vilka hjälpmedel jag använt mig av. Avsnitt tre går igenom implementationen och metoden jag använt. Resultat och slutsatser återfinns i avsnitt fyra respektive fem.

2 Hjälpmedel

I projektet har jag använt mig av en annoterad korpus och en statistisk klassificerare.

2.1 Korpus

Den korpus jag använder är SUC (1), Stockholm Umeå Corpus, som består av 1 miljon ord. För varje ord finns information om böjd form, ordklass och lemma. (se Tabell 1).

Särskilt	ab	särskilt
smygrustningen	nn.utr.sin.def.nom	smygrustning
vad	ha	vad
gäller	vb.prs.akt	gälla
missiler	nn.utr.plu.ind.nom	missil
oroar	vb.prs.akt	oroa
.	mad	.

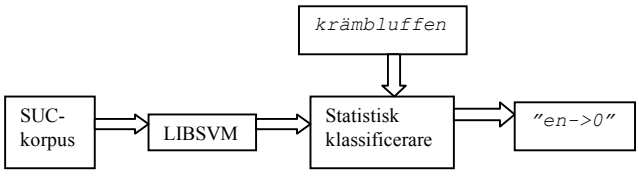
Tabell 1: Utdrag ur SUC-korpus

Texterna i SUC är skrivna under 1990-talet och hämtade från olika genrer.

2.2 Klassificerare

En klassificerare är en funktion som givet ett antal inparametrar ger en klass t ex ”*en->0*” i fallet *krämbluffen*. För att skapa en klassificerare har jag använt mig av programmet LIBSVM (2) och hjälpprogrammet SimpleSVM (3). LIBSVM tar in ett antal exempel, träningsmängd, och använder sig av en statistisk algoritm för att generera en klassificeringsfunktion som passar så bra som möjligt med en träningsmängd. Hjälpprogrammet SimpleSVM förenklar användningen av LIBSVM.

3 Implementation



Figur 1: Översikt över implementationen

Figur 1 visar hur SUC-korpusen, eller rättare sagt delar av den, används som träningsmängd till LIBSVM för att träna en klassificerare. Därefter visas att en klassificerare skapas. Klassificeraren kan sedan användas för att klassificera ett valfritt okänt ord. SUC-korpusen har i projektet även använts som testmängd. I nedanstående avsnitt går jag närmare in på varje steg.

3.1 Träningsmängder och Testmängd

Som träningsmängd och testmängd har jag använt ett blandat urval ur SUC bestående av substantiv, adjektiv och verb. Testmängden består av 50 000 ord. Träningsmängden har bestått av mellan 1000 och 10 000 ord. Följande storlek på träningsmängden har använts:

- 1000 ord
- 2000 ord
- 5000 ord
- 10 000 ord

3.2 Inparametrar

För varje ord i träningsmängden behöver LIBSVM den korrekt klassificerade klassen – för ordet smygrustningen är den ”*en->0*” – och ett antal inparametrar. Jag har använt följande sex inparametrar:

- Ordets suffix upp till fem bokstäver
- Ordets ordklass

På formatet som hjälpprogrammet SimpleSVM använder blir indata för ordet *smygrustningen*:

```
en>0|n|en|gen|ngen|ingen|nn.utr.sin.def.nom
```

3.3 Träna klassificeraren

Orden i träningsmängden skrivs om på ovanstående format och sparas i en fil `data.txt`. Hjälpprogrammet läser in textfilen och omvandlar den till ett numeriskt format (`data.txt.processed`). Dessutom skapas ett kodningsobjekt (`data.encoding`). Jag har tränat klassificeraren med anropet:

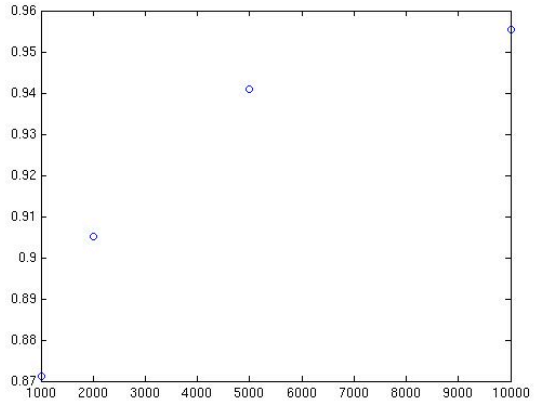
```
svm-train -c 256 -g 0.0025
data.txt.processed data.model
```

som skapar ett modellobjekt (`data.model`).

Kodningsobjektet och modellobjektet behövs när man använder klassificeraren.

4 Resultat

För de fyra olika träningsmängderna har jag mätt andelen korrekt klassificerade ord (se figur 2).



Figur 2: Andel korrekta klassificeringar som funktion av antalet ord i träningsmängden.

Jag har försökt ta reda på vilka grammatiska former som var svårast att klassificera. Eftersom inte alla former förekommer lika ofta i korpusen, varifrån test- och träningsmängderna är tagna, är det svårt att redovisa vilka ord som är svårast att klassificera. Jag har undersökt det fall då träningsmängden bestod av 10 000 ord. I tabell 2 redovisar jag de fem former som fick högst antal klassificeringsfel per antal förekomster i testmängden. Jag tar inte med former som förekom mindre än 70 gånger i testmängden eftersom jag antar att de inte förekommit så ofta i träningsmängden heller.

ordklass	**andel fel**
nn.utr.plu.def.gen	38%
vb.prt.sfo	31%
nn.neu.sin.def.gen	29%
nn.utr.plu.def.nom	28%
nn.neu.sin.def.nom	13%

Tabell 2: Ordklasser som var svåra att klassificera.

Exempel på ord i ordklassen nn.utr.plu.def.nom som blev felklassificerade är *pojkarna* och *gaffeltruckarna*. Dessa ord skulle enligt klassificeraren ha lemmorna *pojkare* respektive *gaffeltruckare* medan de rätta lemmorna är *pojke* respektive *gaffeltruck*.

Ett annat exempel är *barkborrarnas* (nn.utr.plu.def.gen) som enligt klassificeraren skulle ha lemmat *barkborr* men enligt korpusen har lemmat *barkborre*. Dock skulle ju klassificeringen kunna vara korrekt eftersom man kan tänka sig att det finns ett redskap som används för att borra i bark, en barkborr.

Ett sista och positivt exempel är ordet jag tog som exempel i inledningen, *krämbluffen* (nn.utr.sin.def.nom). Detta ord blev korrekt

klassificerat. Så var även fallet för majoriteten av orden i denna ordklass. Bara 4% blev felaktigt klassificerade.

5 Slutsatser och diskussion

Slutsatsen man kan dra av resultaten i figur 2 är att ju fler ord träningsmängden har desto bättre blir klassificeraren. Med en träningsmängd på 10 000 ord blev 96% av orden i testmängden korrekt klassificerade. Kurvan ser ut att plana ut efter detta antal. Jag har inte undersökt resultaten för träningsmängder med fler ord eftersom det tog för lång tid att köra programmet då.

För att metoden ska ge ett helt korrekt resultat skulle man antagligen behöva fler inparametrar till klassificeraren. Ord som *pojkarna* och *gaffeltruckarna* har ju samma ändelser i böjd form men ska klassificeras olika. Dock är det svårt att säga vilka ytterligare inparametrar som behövs. Dessutom visar exemplet med *barkborrarnas* att det inte är möjligt att få en hundraprocentigt korrekt klassificering eftersom det finns ord som kan tolkas på olika sätt.

Slutligen kan man förstås tänka sig andra sätt att angripa problemet. Ett sätt hade kunnat vara att dela upp det okända ordet i kända ord. *Krämbluffen* skulle då delats upp i *kräm* och *bluffen*. Lemmat för *kräm-bluffen* skulle antas ha samma ändelse som lemmat för *bluffen*.

6 Tack

Tack till min handledare Richard Johansson och min föreläsare Pierre Nugues.

Också ett tack till min vän och diskussionspartner Kajsa Lindén.

Referenser

1. Eva Ejerhed, Gunnel Källgren, Ola Wennstedt och Magnus Åström: "The Linguistic Annotation System of the Stockholm-Umeå Project", 1992.
2. Chih-Chung Chang and Chih-Jen Lin: "LIBSVM: A Library for Support Vector Machines", 2001.
3. Richard Johansson: "SimpleSVM: A Java-based Wrapper Library for LibSVM". Software available at http://www.df.lth.se/~richardj/simplesvm.