Tagging of named entities in Swedish traffic accident reports

Mats Svensson
Enea Systems AB
Box 4150
203 12 Malmö, Sweden
masv@enea.se

Abstract

A system has been designed to tag named entities (NEs) from text. The relevant domain is traffic accident reports. The texts are written in the Swedish language. The NEs to be tagged are names of roads, streets, city squares, towns and cities.

The system makes use of a rules-based approach. Gazetteers are used to find larger cities, morphological rules are applied to individual words, and context rules are applied to groups of words.

The project has shown evidence that the formation of Swedish words aids in identification and tagging of information in text.

1 Introduction

The task was to develop a system to be incorporated into the CarSim project [1]. For this purpose, a sub-system that can tag towns, cities, roads, highways, streets and city squares was needed.

The task at hand is divided into two parts:

- Detection of roads, highways streets and squares.
- Detection of city and town names.

It turned out that detection of roads, squares and particularly highways can be done with very good results by applying simple regular expressions to text. However, the formation of town names is a lot more complex than the formation of street names. This is due to the fact that many town names are composed of word stems belonging to a much older form of the language in use today.

The working hypothesis at the outset of the project was to use regular expressions to perform the first task, and to use morphological rules combined with context rules to solve the second problem.

2 Related work

This project has been inspired by the work of Nugues et al [1]. Another source of inspiration is the tagging systems developed by Andrei Mikheev and his team described in [4].

3 Regular expressions

In certain parts of this text, regular expressions are used. The syntax used is standard, however a few notations may need a further explanation.

A typical regular expression to match the letters A to Z may be written as [A-Z]. In Swedish, the alphabet is expanded with three more letters after Z, the letters Å Ä and Ö. In some regular expressions in this text, the patterns [A-Ö] and [a-ö] appear, indicating a pattern that would match all characters of the Swedish alphabet.

4 Tags

In order for the CarSim system to be able to operate on the NEs found in the text, these are tagged with XML tags. The XML tag set used was invented for this purpose, but follows the MUC[1] ENAMEX convention. The following tags are used by the system:

<ENAMEX TYPE="ROAD">	Country roads.
<ENAMEX TYPE="HIGHWAY">	Highways.

[1] Message Understanding Conferences. See [2].

<ENAMEX TYPE="STREET">	City streets.
<ENAMEX TYPE="SQUARE">	City squares.
<ENAMEX TYPE="CITY">	Major city.
<ENAMEX TYPE="TOWN">	Towns and minor communities.

The tags are applied to text as follows by this example:

Olyckan skedde på <ENAMEX TYPE="ROAD"> *väg 16*</ENAMEX> *mellan* <ENAMEX TYPE="TOWN"> *Dalby* </ENAMEX> *och* <ENAMEX TYPE="CITY"> *Lund*</ENAMEX>*. Lyckligtvis, för de inblandade, befann sig en polispatrull i närheten på* <ENAMEX TYPE="STREET"> *Norrängavägen* </ENAMEX>*.*

Translation: The accident occurred on road 16 between Dalby and Lund. Fortunately for those involved, a police patrol happened to be nearby on Norrängavägen[2]*.*

5 Order of processing

The system goes through a number of processing steps in sequence to perform the tagging. These steps are:

1) Build up an internal data representation of the text for efficient access to tokens.
2) Perform a pass through the text looking for and tagging county and country roads and highways.
3) Perform a pass through the text looking for and tagging compound street names.
4) Perform a pass through the text looking for and tagging multi-token street names.
5) Use a gazetteer to find and tag the names of major cities.
6) Perform a pass through the text to tag towns based on word prefix and suffix.
7) Perform a pass through the text to find towns by using context rules. In this step, a list is built of possible towns.
8) Pass through the list of possible towns and clean out all duplicates and known false hits.
9) Use list of towns as a gazetteer to tag the text for towns in a final pass.
10) Print out tagged representation of text.

At each tagging stage, previously tagged groups of tokens are ignored by the system in order to speed up processing.

6 Finding roads and highways

Swedish county and country roads as well as highways follow very simple rules of naming. This makes detecting them quite trivial.

The common way for country and county roads to be mentioned is as *väg 16*, *länsväg 16* or *riksväg 108* with the possible abbreviations *lv 16* and *rv 108* for the last two[3].

These can be found by looking for the following patterns in the text:

väg **Nn**
väg **Nnn**
riksväg/rv **Nn**
riksväg/rv **Nnn**
länsväg/lv **Nn**
länsväg/lv **Nnn**

Here, **N** denotes a digit from 1 to 9 and **n** denotes a digit from 0 to 9. The slash denotes a choice between any of the words it separates.

Finding Swedish highways is similar to the task of finding roads. Swedish highways are named *E6*, *E22* and the like, an *E* followed by one or two digits, where the first one is always non-zero.

The highway tagger looks for the following patterns:

E**N**	E **Nn**
E**nn**	E**N**:an
motorväg E**N**	E**Nn**:an
morорväg E**Nn**	E **N**:an
Väg E**N**	E **Nn**:an
Väg E**Nn**	
E **N**	

[2] *Dalby* is a minor town, *Lund* is a city and *Norrängavägen* is the name of a street.

[3] *väg 16* means *road 16*, *länsväg 16* means *county road 16* and *riksväg 108* means *country road 108*. The abbreviations are more common in police reports and are never used in newspaper text.

These patterns have proven to be adequate for tagging all roads and highways in text taken from the relevant domain. They achieved a 100% precision and 100% recall score in a test where a 290 kB development corpus of relevant domain-specific text was tagged[4].

7 Finding compound street names

The Swedish language very often uses compounding to form new words, quite the opposite to what is done in Romance languages and English.

An example is a person who works with cleaning windows. He is referred to as a *window cleaner* in English, but in Swedish, the word *fönsterputsare* is formed by the words *fönster* (window) and *putsa* (to clean or to polish something).

This compounding property of the language also reflects how city streets and town squares are named. A typical Swedish street may be named after an entity compounded with a word for a street.

For example a street called *Oak street*, named after the oak tree would be called something like *Ekvägen* in Swedish, where *Ek* is Swedish for the oak tree or oak wood, and *vägen* is the nominative form of a word meaning *street*.

The compound street tagger uses this regularity of the language to find roads and city squares. The following regular expressions are used to search for target expressions:

[A-Ö][a-ö]*gatan	[A-Ö][a-ö]*vägen
[A-Ö][a-ö]*stigen	[A-Ö][a-ö]*torget
[A-Ö][a-ö]*platsen	[A-Ö][a-ö]*gränd
[A-Ö][a-ö]*gränden	[A-Ö][a-ö]*leden

These regular expressions may of course score hits that prove to be wrong. A good example is that we want to be able to detect and tag *Götaplatsen*, a square in the city of Göteborg. However, within the domain, a very common word is *olycksplatsen* (the scene of the accident). When this starts a sentence, the word is capitalized, which will cause a false hit.

In order to get around this, the system uses a gazetteer of false hits, an exclusion list. Tokens or groups of tokens are compared against exclusion lists to avoid tagging words such as *Olycksplatsen*.

Of course, each such exclusion list is both specific to the domain and specific to the tagging task at hand. The contents of such a list along with the design of the regular expressions that it guards can be seen as the training of the system.

8 Other street names

Compounded words are not the only way to form street names in Swedish. A very small fraction of street names are formed completely irregularly, but most still contain words like *väg, gata* (road, street), *stig* (path) and so on.

An extension to the notion of compound street names is multi-token street and square names, henceforth referred to as just multi-token names.

A multi-token name is a street such as *Ernst Wigforss gata* (a street in Lund), *Lilla Torg* (a square in Malmö) and so on, composed of several tokens.

These are easy to detect by using regular expressions and exclusion lists. These are some of the rules used to find such streets:

[A-Ö][a-ö]*	[A-Ö][a-ö]*sgata	[A-Ö][a-ö]*s gata
[A-Ö][a-ö]*	[A-Ö][a-ö]*sgatan	[A-Ö][a-ö]*s gränd
[A-Ö][a-ö]*	[A-Ö][a-ö]*sgränd	
[A-Ö][a-ö]*	[A-Ö][a-ö]*sväg	
[A-Ö][a-ö]*	[A-Ö][a-ö]*s väg	
[A-Ö][a-ö]*	[A-Ö][a-ö]*s gata	

Examples of exclusion list entries for these rules are expressions such as *På väg* (on the way), *Hans väg* (His path) and similar, common constructs of the language that deal with the word *väg* (road, path).

Similar rules work fine for squares, again keeping in mind that *plats* (a synonym for square in Swedish) is also a common word meaning *place* or

[4] The test corpus consisted of press clippings on car accidents from Swedish news sources.

location, which has to be kept in mind when designing the exclusion list for this rule set.

9 Tagging major cities

The system uses a gazetteer to find those tokens that are to be tagged as CITY. The reason for this is that there are not many major cities in Sweden. The test system developed used a gazetteer of 22 cities for this purpose.

The system will look for two things when using the gazetteer. Not only is the exact match to the listed item sought, but also the word with the letter 's' appended to form the possessive form of the word.

The system will find the city *Stockholm* if an exact match can be done or a match can be done to the variant *Stockholms*.

Swedish is abundant with compounded words, as has already been noted. A phrase such as *the police in Stockholm* is built up as one word, *Stockholmspolisen* in Swedish. The system will not, and shall not tag such compounds due to the fact that they seldom refer directly to a location but rather to where a certain subject is from.

10 Tagging towns using prefixes and suffixes

As in many other European languages, Swedish town names often stem from words that were in common usage hundreds of years ago, creating patterns in their formation. This, combined with the Swedish quirk for compounding words, is very useful when it comes to searching for town names.

Although there are local themes and variations on formation of town names, most notably far north, where naming often stems from the Saami language rather than from Swedish, morphology can be used to detect town names. Most notable in Swedish town name formation is the plethora of common prefixes and suffixes.

The system works with two lists of suffixes and prefixes, which are matched against capitalized words in the text. This is combined with exclusion lists containing common words that may be formed along the same rules.

Some examples of common suffixes used are:

-arp	-by
-stad	-fors
-boda	-vik
-sala	-järvi

These are present in town and city names such as *Genarp*, *Hyby*, *Filipstad*, *Bengtsfors*, *Lönsboda*, *Valdemarsvik*, *Onsala* and *Jukkasjärvi*.

When it comes to prefixes, very many of the more common prefixes involve names of kings or other historical persons as exemplified by the cities *Karlstad* and *Karlskrona*.

However, it turned out that the suffix proved to be a better source for identification of a town name than the prefix. The final system looks only for these prefixes:

Mal-	Kung-
Kristian-	Karl-
Chistian-	Carl-
Näs-	Berg-
Kal-	

It is quite possible to find a lot more common prefixes on Swedish towns, however, extending the prefix list when using a large suffix list added very little to the detection scores of the program.

One large cause for false hits by using the suffix rules was that very many surnames of people are formed in the same way as town names. *Lindesberg* is a town, whereas *Magdalena Forsberg* is a person. A further complication is that the target domain consists of traffic accident reports from newspapers and similar sources, where people often are introduced once with full name, and thereafter mentioned only by last name.

To avoid scoring false hits due to this kind of complications, the system tries to determine whether a name, rather than a town, has been

found. If a possible town is preceeded by a capitalized word, this word is checked against a list of common first names. If there is a match, the word is considered to be a name rather than a town.

Another method used to find names is based on the Swedish tradition of double names. Double names in Swedish are first names that are hyphenated, like *Jan-Åke* and *Anna-Karin*. If a possible town is preceeded by two capitalized words, which are separated by a hyphen, it is also considered to be a name.

If a word has been categorized as being a name and not a town, a backwards search is done about 50 words, and a forward search about 250 words to remove false tagging of the name. The reason for this is that it is very common for newspaper text to omit the first name of a person in the headline, introduce the person by full name somewhere near the beginning of the text and then refer to the person by last name only for about one quarter of a page until the full name is mentioned again.

11 Tagging towns using context rules

Very many Swedish towns can be found by the simple rules described above. However, there are other town names that are not as regularly formed. Whereas *Bjärred* can be found by the suffix rules and *Kungsör* can be detected by the prefix rules, the town of *Lomma*, which is close to Bjärred geographically, will not be detected. None of the towns *Morgongåva*, *Virke* or *Råå* would be found by those rules either.

In order to improve detection, a set of context rules was introduced. These are used to analyze the surroundings of a word. In the domain of traffic accident reports, town names are commonly used in certain special contexts.

The context rules are applied by scanning the text for matching patterns. As soon as a complete match is made, the words that may possibly be towns are extracted. These are then added to a list of town candidates for further processing.

These are examples of context rules[5]:

polisen i **TOWN1**
ROAD1 mellan **TOWN1** och **TOWN2**
norr om **TOWN1**
trafikolycka i **TOWN1**
bilolycka i **TOWN1**
STREET1 i norra **TOWN1**

Note the type tags: ROAD1 will match any token or group of tokens tagged as a road by the earlier passes that the system made over the text. Also, STREET1 will match anything tagged as a street.

After the list of town candidates has been built, it is subject to elimination of duplicates. Any town found should only be on the list once.

After this, all already known cities are eliminated, as are all towns that match known prefixes and suffixes. Exclusion lists are also applied, both on suffixes and on entire words, to remove references to gas stations, forests, streets, sports arenas and so on. The reason that the exclusion lists operate on suffixes is the affinity for compounding words in Swedish.

As an example, we want to find the town *Virke* in this text: "*Olyckan inträffade på väg 104 mellan Virke och Stora Harrie. Mer exakt inträffade olyckan strax norr om Shellmacken.*"[6]. We do not, however, want to get the word *Shellmacken* on the towns list[7].

The final result of the application of context rules and post-processing is a list of towns. This list is used to make a pass through the text and tag all occurrences of those words.

[5] The translations are The police in TOWN1, ROAD1 between TOWN1 and TOWN2, North of TOWN1, Traffic accident in TOWN1, Car accident in TOWN1 and STREET1 in northern TOWN1.

[6] *The accident occurred on road 104 between Virke and Stora Harrie. More specifically, it happened west of the Shell gas station*. This sentence is, however, constructed. There are no gas stations in Virke, and road 104 does not run through the town.

[7] The word means *the Shell gas station,* and is another example of compounding of words in the Swedish language.

A test on the 290 kB development corpus showed that 50 towns that could not otherwise be detected were found by this set of rules, of which 11 were false hits. All false hits were actual geographic locations, however. Some were lakes and islands, some were foreign countries and some were city boroughs[8].

12 Multi-token towns

Some towns in Swedish are augmented with extra, descriptive words. Examples are *Lilla Edet*, *Stora Råby* and *Södra Sandby*. Common for many of these is that the first word is almost always an adjective like *Norra* (northern) or *Lilla* (lesser).

Therefore, the system is instructed to look for a list of adjectives before any towns that are found. If a capitalized adjective that is on the list is found before the town name, this is also incorporated in the tag.

13 Results

The final system was blind-tested on texts it had not been subject to before. A 14 kB text was used, which was composed of domain-specific text from Swedish news sources. The total number of tokens in the text (words and punctuation) was 2533.

On this text, precision and recall was measured. Recall is measured as the number of relevant results in the answer set over the total number of possibly relevant results. Precision is measured as the ratio of relevant results over all results in the answer set.

For the test text, a recall of 93% and precision of 89% was measured. Upon inspection, the text showed evidence that the results could be improved by adding more context rules and restricting more words by using the exclusion lists for town suffixes[9].

14 Conclusions

The project has shown that detection and tagging of Swedish towns, streets, roads and cities can be done quite reliably by using a rules-based approach. Evidence was found that roads and highways can be automatically detected with absolute confidence by applying simple rules based on regular expressions. City squares and streets may also be detected quite reliably by applying slightly more advanced regular expressions.

When it comes to detecting and tagging towns, the approach that worked best was to look for common suffixes. Prefixes added a bit more recall (it went up from 89% to 93%), and precision was helped by the addition of the exclusion lists (an improvement from 81% to 89%).

The towns that were not detectable by use of prefix and suffix rules could in some cases be found by the context rules instead. It turned out that the few simple context rules used for the test system worked well enough to tag only geographical locations, however, only about 80% of the locations they managed to detect were actually towns.

References

[1] Per Andersson, 2003, *A Prototype to Extract and Visualize Information from Car Accident Reports in Swedish*. Master's Thesis, department of Computer Science, Lund Institute of Technology.

[2]**WEB:**

http://www.cs.nyu.edu/cs/faculty/grishman/muc6.html

[3] Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel et al., 1996, *FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural Language text* in Roche and Schabes, eds., *Finite State Devices for Natural Language Processing*, MIT Press, Cambridge MA, 1996

[4] Andrei Mikheev, Marc Moens and Claire Grover, 1999, *Named Entity recognition without gazetteers* in *EACL'99, Bergen, Norway* ACL June 1999. pp. 1-8

[8] It may, with good reason, be possible to argue that a city borough is a town.

[9] It shall be kept in mind that the system, as described here, is a small prototype system.