Extracting Information from Sport Articles in Swedish Using Pattern Recognition

Erik Lindvall
Lund University
f98el@efd.lth.se

Johan Nilsson
Lund University
f98jn@efd.lth.se

Abstract

Sport articles from newspapers containing end results of one or multiple games is a special application of the information extraction task, in that almost all interesting information is available in patterns which are easy to recognize. Use of pattern recognition to extract this information yields good results especicially considering the simple implementation. Such a sytem has been implemented and tested for Swedish and the results are considered satisfactory.

1 Introduction

The subject of this report is to describe a method to extract information from sports articles in Swedish taken from the webpages of various Swedish newspapers. A system using this method has been implemented and the results of this implementation are discussed in the report as well. The idea has been to utilize specialized tools for extracting information which is known to be available in a specific form.

Our strategy for extracting information has been a very ”shallow” one, focusing wholly on obtaining the relevant information and not on constructing a full grammar which would result in a deeper understanding of the texts. The articles typically contains a lot of irrelevant information which in this application is not interesting. A shallow approach will save time by skipping processing of this information.

Some improvements to our method are suggested, as well as a brief discussion on some alternative approaches to achieve the same end. A small dictionary, containing English translations of the Swedish terms used in this report, can be found in appendix C.

1.1 Interesting Information

The application focuses on sport articles which are basically reports from football or ice hockey games in Swedish or foreign leagues. From these reports we wish to extract the following information from every game mentioned in the article:

1. The team names
2. The end result, i.e. which team (if any) won the game
3. The final score

This is information which as good as certainly is available in some form in the article.

In an improved version the program could include detection of for instance goalscorers, home/away team, referee or even which sport the article is about. This information is not certain to be included in the article however (except for the sport which in most cases is very implicit), which raises some questions when it comes to evaluating the program. Our first version has been focused on extracting information known to be available for simplicity in both implementation and evaluation.

2 Method

2.1 Basic Strategy

The basic strategy for obtaining the information is simple pattern recognition. The first step of the algorithm will detect key patterns, such as scores, team names and certain ”winning” or ”losing” words. The output from this step will undergo some local (meaning same sentence) processing to get a standardized output from every sentence.

The output from the sentences in the pattern recognition-step are completed by a global logic which will take a larger discourse into account to improve the output from the local logic when this logic is incomplete. For example, the output from a sentence may consist in something equivalent to ”team1 loses”. The global logic will in this case try to determine the winner of that specific game from the larger discourse.

The output from the global logic will consist in a number of *facts* with a given priority, which will undergo further processing in the final step. In this final cleanup step some facts are absorbed into others and other facts are outright discarded, based on the priority every fact received in the last step. For example, the fact ”team1 wins” can be absorbed into the fact ”team1 defeats team2”, and if this fact has a high priority it can cause the fact ”team1 loses” to be discarded.

When this is done a list of facts and their respective priorities will remain. This list will not include facts which are inconsistent with each other, and no two facts which say the same thing. The output will consist of all facts that have a priority higher than a certain threshold, which will be tuned to give the best possible result.

2.2 Initial Pattern Recognition

The first step is to detect the interesting patterns. The patterns which are detected have basically four different forms: the *team*-pattern, the *score*-pattern, the *win/lose*-pattern and the *draw*-pattern.

One of these patterns, the win/lose-pattern, is more complicated than the other three. The score- and team-patterns need some explanation as well. The draw-pattern simply exist of a draw-key, which in this version only can consist in one word: ”oavgjort” and its different forms. For more discussion on the draw pattern, see 3.2.1. The other patterns are described below in more detail.

2.2.1 The Team-pattern

The team-pattern is simply a team name, which can consist of a prefix, a *location* and a suffix (although most of the times not with both prefix and suffix), solely a prefix/suffux or solely a location. An example of a team with a prefix and a location is ”IFK Göteborg”. An example of a team with a location and a suffix is ”Örgryte IS”.

The team-pattern is very similar to a multiword since the prefix/suffix must come right before/after the location in the text for the pattern to be recognized. However, one more term is included in the team-pattern, since it has crucial importance for in what context the team appears in the text: if there is a preposition of a special kind in front of the team this is included in the pattern.

The prepositions we are interested in are six in number: ”av”, ”till”, ”för”, ”på”, ”över” and ”mot”. We want to save these preposition for further processing because of their implications on the team following them in the text.

2.2.2 The Score-pattern

The score pattern can also be seen as a kind of multiword. It consists in a construction on the form `Number1 - Number2`, where `Number1` and `Number2` are natural numbers separated by a dash. This is the form which is used for reporting game scores.

2.2.3 The Win/lose-pattern

This is the most complicated of the basic patterns, and is different in that it cannot be reduced to a multiword. This pattern is on the form

```
Team1 [...] Key [...]
   Team2 [...] Team3
```

where `Team1`, `Team2` and `Team3` are team-patterns and `Key` is a word which says something about the result of the game. The ellipses are meant to represent the fact that there can be words, which are irrelevant for our application, between the parts of the pattern.

Further, a maximum of 2 teams are detected in every pattern (see below). As an example, the sentence “IFK Göteborg lyckades till slut besegra

ett svagt Örgryte efter en spännande match.” will match the pattern above with `Team1` = ”IFK Göteborg”, `Key` = ”besegra” and `Team2` = ”Örgryte”. `Team3` will be unassigned in this case.

There are also different types of patterns depending on what the key-word is. Keys can be *active* or *passive* and *winning* or *losing* words, generating a total of four different patterns. The distinction between winning and losing keys is obviously important since we are interested in who won, and the active/passive distinction is important for the same reason due to its implications on the word order.

2.2.4 Active/passive Distinction

Most key-words are active. This includes verbs in active form such as ”vinna” and ”förlora”, certain nouns such as ”förlust” and ”seger” and even some verbs in passive form such as ”slogs”. The reason for the last ones to be included is purely empirical - we found that if the last example is considered active rather than passive the results improve. This is also rather uncontroversial if one takes into account the symmetry of winning and losing respectively.

The sentences ”Göteborg besegrade AIK” and ”Göteborg besegrades av AIK” are similar for our purposes but need to yield opposite results. It seems logical to count ”besegrades” as an active losing verb even though this will complicate the terminology. In fact, in the current version only two constructions are considered passive: ”vinnas” in different tenses and ”förloras” in different tenses.

Note the difference between ”Matchen förlorades av AIK.”, which will match `Key` = ”förlorades”, `Team2` = ”AIK” and ”IFK besegrades av AIK” which will match `Team1` = ”IFK”, `Key` = ”besegrades”, `Team2` = ”AIK”. In the first example AIK is the loser and in the second the winner. These need to be separated, and since our detection of the winner is based solely on the key-word and the word order, it is necessary to take active/passive distinction into account.

2.2.5 Conflicting Patterns and Priorities

Sometimes a sentence will be ambigous in the sense that it can match multiple patterns. Consider for example ”IFK besegrade AIK som tidigare i veckan hade vunnit mot Örgryte.” In this sentence we can match (among others) `Team1` = ”IFK”, `Key` = ”besegrade”, `Team2` = ”AIK” or `Key` = ”besegrade”, `Team2` = ”AIK”, `Team3` = ”Örgryte”. It is important to have a clear priority-order so it is known which patterns are matched and which are discarded. The order of matching attempts goes as follows:

1. Team1 Key Team2
2. Team1 Key
3. Key Team2 Team3
4. Key Team2
5. Key

In other words, we try to fill as many slots as possible from the beginning of a sentence. In the example above the first matching mentioned would be the one detected. Note that if no teams are detected a key-word will still generate a pattern. In later steps this key can be connected to teams mentioned in the current discourse (see 2.4.1).

2.2.6 Output

Output from the first step will include, for every sentence:

1. A list of teams mentioned in the sentence
2. A list of scores mentioned in the sentence
3. A list of win/lose- and/or draw-patterns detected in the sentence

2.3 Local Logic

The output from the initial pattern recognition is taken as input to the second step, which is a local logic for extracting information which is available in every sentence. In this step the active/passive distinction and the win/lose keys disappear, and the result is simply something like ”IFK wins” or ”IFK defeats AIK”. Depending on the key-variable in every fact (which can be of four types: **win_act**, **win_ps**, **lose_act**, **lose_ps**) and the word order, new facts like the ones above are obtained. A more detailed specification of this process can be found in appendix A.

2.3.1 Prepositions Acting on Teams

In this step further processing of the teams is also done. A crude division of the prepositions into *subject* and *object* prepositions is made in an attempt to decide in what context the team appears. It is important to note that the terms subject and object here (and more importantly in the next step) does not appear in their grammatical senses. In our terminology a subject is a team which the text is ”about”, and an object is a team which is mentioned relative to the subject.

Prepositions which appear before the teams can be a clue to whether the team is a subject or an object in this sense, which is why these are labelled subject and object preposition. For example, ”mot” is a typical object preposition. If the pattern ”mot IFK” appears in a sentence it is reasonable to believe that this sentence is about another team which played against IFK. IFK is thus an object in this sentence since it appears relative to another team. Our crude division of preposition identifies subject prepositions as ”till”, ”för” and ”av” and object preposition as ”mot”, ”över” and ”på”.

2.3.2 Output

The output from the local logic will include, for every sentence:

1. A list of teams and whether they are preceeded by subject prepositions, object prepositions or no prepositions.

2. A list of facts on the form ”Team1 defeats Team2”, ”Team1 wins”, ”Team1 loses”, ”win” or ”lose”

3. A list of scores unchanged from the previous step

2.4 Global Logic

2.4.1 Discourses

When processing language on a discourse level it is usual to keep track of a number of *discourse entities*, which are terms that the text is ”about” and refer to implicitly (see e.g. Nugues (2002)).

Our approach can be compared to the one above. The discourse in our application is simply one team which we assume the text to be ”about”. After studying game reports of the kind we wanted to process we found that usually the text was about one team primarily, with comments on what team was the opponent of the main team, what the score was etc. Sometimes the main team changed during the text. To reflect this, a ”subject” for every sentence is determined during the global step in an attempt to decide the main team for that sentence.

2.4.2 Finding the Subject

The subject of a sentence is determined as follows (and in this order):

1. If a sentence contains a team which is not an object, and the same team is the subject in the preceeding sentence, the team is set as subject in the current sentence.

2. Else, if a sentence contains a team preceeded by a subject preposition the team is set as subject in the sentence.

3. Else, if a sentence contains a team not preceeded by any preposition the team is set as subject in the sentence.

4. Else the subject of the current sentence is set equal to the subject in the preceeding sentence.

5. If the first sentence does not contain a team its subject is considered not assigned or undetermined.

All sentences except possibly some initial ones are thus assigned a subject.

2.4.3 Completing Facts

After the subjects are determined they are used to complete some facts which are incomplete (for example ”Team1 loses”). A priority is also assigned here depending on how certain the fact is judged to be. For a detailed reference on how and in which order the facts are completed and what patterns give what priorities see appendix B. A higher priority means a more certain fact.

2.4.4 Assigning Scores to Facts

Once the facts are completed an attempt to assign scores to all facts is made. The score is also assigned a priority in a simliar manner as above. Three different priorities can be set: 0, 1 and 3.

For every fact, all sentences are checked for scores, and the score with the highest priority found is set as score for the fact. Score priority is as follows when matching the pattern `[Team1 defeats Team2]` against a sentence:

- If the list of teams of the sentence includes either `Team1` or `Team2` and the score-list of the sentence is non-empty, a member of the score-list is set as score for the current fact with a priority of 3.
- If the subject of the sentence is either `Team1` or `Team2` and the score-list of the sentence is non-empty, a member of the score-list is set as score for the current fact with a priority of 1.
- Else, the score is considered unassigned and has a priority of 0.

A possible complication here is the case when the score-list of a sentence contains more than one entry. This rarely happens, but a better approach which will take only a minor improvement of the system would be to choose the highest score from the score-list instead of just an arbitrary one.

2.4.5 Output

The global logic will have a list of facts and their scores with priorities assigned both to the fact and the score as output.

2.5 Cleanup

In the final step the list of facts is processed and some facts are removed before the final output. First, a combined priority is assigned, which is the priority for both the fact and the score in one. This is calculated from the priorities of fact and scores as $10 \cdot$ (fact priority) $+$ (score priority) to reflect that it is more important for the facts to be certain. The cleanup is then done in two steps.

2.5.1 Unification

In this step facts are compared and if possibly unified. This is done by taking the fact with highest combined priority and comparing it with all other facts. This will create a list of new facts whose priority is set to the priority of the generating fact. After this, the fact with second highest priority will unify with the (remaining) list and so on until the list is empty. For example, given the list of facts:

```
[IFK, wins, noopp]
[AIK, loses, noopp]
[MIF, loses, noopp]
[IFK, wins, AIK]
```

the first step will unify the list to:

```
[IFK, wins, AIK]
[IFK, wins, MIF]
```

given that the first fact has the highest priority. Note that for this to happen, the different facts cannot have different scores.

2.5.2 Inconsistencies

In the second step entries which are not consistent are deleted from the list. Two entries are considered inconsistent if the teams are the same but the result is different. In such a case the entry with lowest priority is deleted from the list of facts.

2.5.3 Priority Threshold

Finally, a threshold is set and all entries with priorities below this threshold are deleted. This is done to allow tuning of the sytem in a simple way. In our application the threshold is set 0.

3 Results

The output from the system is a list of games (with teams, end result and final score), for instance

```
[[['IFK', 'Göteborg'], wins,
['AIK'], [2,'-',1]],
[['Djurgården'], draws,
['Halmstad'], [1,'-',1]]]
```

In this example two games were found, IFK Göteborg vs. AIK and Djurgården vs. Halmstad.

Each entry in the list of games consists of four fields:

```
[First_team, wins|draws,
Second_team, Score]
```

One of `First_team` and `Second_team` (but not both) can be `noopp` when no opponent is found. Also, `Score` can be `noscore` when no score is found.

3.1 Method of Scoring

The scoring is done as follows: for each game entry in the output, find an entry in the template that matches the game considered, i.e. find an entry in the template containing at least one of the teams in the considered output entry.

If several matchings are possible, the one resulting in the highest score is used. Note that `noopp` and `noscore` fields are counted as empty, i.e. they are not considered as irrelevant when calculating the precision.

The first field in an entry is considered correct if it matches the first or third field in the corresponing entry in the pre-filled template.

For the second field to be considered correct it is required that it matches the second field in the pre-filled entry. In the case of `wins` it is also required that the first and/or the second fields in the two entries match (i.e. the second field is incorrect for instance when the entry is `[T1, wins, T2, S]` and the matching pre-filled entry is `[T2, wins, T1, S]`).

The second team (third field) is correct if it matches the first or second correct team (see above).

Finally, the score is correct if it matches the score in the pre-filled template. Note that it is also correct when it is reversed. i.e. `[2,'-',1]` matches `[1,'-',2]` since it is obvious from field 2 which team won.

Consider

```
[['IFK', 'Göteborg'], wins,
['AIK'], [2,'-',1]]
```

vs. the correct

```
[['AIK'], wins, ['IFK',
'Göteborg'], [2,'-',1]]
```

Field 1,3, and 4 are correct while field 2 is incorrect.

Now consider the output

```
[[['IFK', 'Göteborg'], wins,
['AIK'], [2,'-',1],
[['Djurgården'], draws,
['Halmstad'], [2,'-',2]]]
```

vs.

```
[[['AIK'], wins, ['IFK',
'Göteborg'], [2,'-',1]],
[['AIK'], draws, ['Elfsborg'],
[1,'-',1]]]
```

The second entry in the output has no match in the correct results, and the second entry in the correct results is not represented in the output. In the first entry of the output, three of the four fields are correct. Hence both recall and precision are $3/8$ is this example.

3.2 Scores

Tested on the corpus consisting of 45 articles from Aftonbladet, 20 articles from Dagens Nyheter, 14 articles from Expressen and 8 articles from Sydsvenska Dagbladet (i.e. 87 articles) the system achieves a recall of 56% and and a precision of 66%.

Compared to the numbers FASTUS achieved in MUC-6 (recall 44% and precision 61%) (Appelt et al., 1993) our result seems very good. However, since the task is most likely much easier than the one in MUC-6, it is not unreasonable to believe that a FASTUS-like system would perform considerably better than ours. Even a very simple heuristic, which ouput the two most common teams in the text and the highest score, would probably perform decently.

On articles where only one game is mentioned (which are quite common in the corpus), the system scores even better that the above figures, with recall of approximately 75% and precision of approximately 85%. As expected, the performance gets worse when the number of teams and games mentioned in an article increases.

3.2.1 The Problem with Draws

It is noteworthy that the system performs considerably weaker on articles containing a game ending in a draw. Especially the recall (36%) is much worse when only articles mentioning a draw are considered. As noted above, our draw-pattern is very simple, which results in this rather weak performance.

However, it is not quite obvious how to construct more efficient draw-patterns; it seems like a draw is much more often expressed in an indirect way than a win or a loss. One way of detecting draws would of course be to utilize scores (e.g. ”2-2” implies a draw), but it is often not very easy to tie the right score to the right game, and to make

sure it is final, when there are several games mentioned in an article.

4 Alternate Solutions and Further Development

4.1 Full Grammar

It is of course possible to construct a complete grammar for the text from knowledge of the Swedish language in an attempt to generate a more full understanding of the text. From this full understanding the pieces of interesting information could then be extracted. To do this a parts-of-speech tagger whose output was inserted into (for example) a DCG grammar could have been used. After this had been completed references on the local and global scale would be worked out.

This approach would take a lot of work and considerably more time to execute, and it is our opinion that it would be difficult to significantly improve the results with this kind of system. The final steps would have to be very accurate to localize the interesting information and a lot of work would be done on text that is not interesting for the application.

4.2 Local Grammar

A better approach is to construct only a local grammar. This could probably improve the system if it was done properly. With a good parts-of-speech tagger it would be possible to make a grammar specifically suited for this application. This was in fact our first approach, but initial results were disappointing (particularly due to bad PoS-tagging). However, a grammar of this kind would help in that it would be possible to more thoroughly investigate in what context the teams appear, and not completely rely on preceeding prepositions and word order.

For a more difficult task (like detecting goalscorers or more) the grammar would probably significantly improve the results. For the limited task investigated in this report the results would most likely be a small improvement.

4.3 Why Pattern Recognition?

There are two factors which make pattern recognition a good approach to choose: the fact that the patterns (such as teams and scores) are easy to recognize, and the rather specialized topics of the articles. Often, the reader of an article is expected to know things not explicit in the article (such as the result of past games). References to such knowledge is equally unattainable for both pattern recognition and a local (or global) grammar. It then seems reasonable to regard this information as ”lost” and chose the simpler approach. The alternative would be to have a large database of past events, but that task is on a whole other scale.

4.4 Improvements in Current Program

A very crude but probably effective improvement would be to do a simple check before the output to make sure a game between the two most common teams in the article is included. In most cases the article is about primarily one game with perhaps some other games mentioned briefly. If these brief interludes affect the output so that the primary game is lost (which in some cases happened), we would wish to make at least a crude guess of a game between these two teams.

Other improvements would be to add patterns, specifically to better detect draw games. This is a matter of finding suitable patterns and adding them to the code. More dramatic improvements would be along the lines mentioned above, i.e. constructing a local grammar for sentences.

References

D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Meyers, and M. Tyson. 1993. SRI International FASTUS system: MUC-6 test results and analysis. In *Proceedings of 16th MUC*, Columbia, MD.

Pierre Nugues. 2002. Manuscript. Department of Computer Science, Lund University. Used as lecture notes in the course ”Introduction to Language Processing and Computational Linguistics” 2002.

A Local Logic

Below is a table which describes how the patterns from the initial steps translates into local ”facts”. The higher up in this table a match occurs, the higher priority it has (i.e. tests for matching are made top to bottom of the table). ObjectPrep is an object preposition, win_act, win_ps, lose_act and lose_ps are active/passive, winning/losing keys respectively and a TeamClause is a team possibly preceeded by a preposition. Team1 etc are teams and in the case where Team1 and TeamClause1 appears in the same entry Team1 is the team contained in the team clause (the same goes for Team2 and TeamClause 2 etc).

Pattern	Translated into
[draw]	[draw]
[[ObjectPrep, Team1], win_act, TeamClause2]	[Team2, defeats, Team1]
[[ObjectPrep, Team1], lose_act, TeamClause2]	[Team1, defeats, Team2]
[[ObjectPrep, Team1], win_act]	[Team1, loses]
[[ObjectPrep, Team1], lose_act]	[Team1, wins]
[TeamClause1, win_act, TeamClause2]	[Team1, defeats, Team2]
[TeamClause1, lose_act, TeamClause2]	[Team2, defeats, Team1]
[win_act, [ObjectPrep, Team1], TeamClause2]	[Team2, defeats, Team1]
[lose_act, [ObjectPrep, Team1], TeamClause2]	[Team1, defeats, Team2]
[win_act, TeamClause1, TeamClause2]	[Team1, defeats, Team2]
[lose_act, TeamClause1, TeamClause2]	[Team2, defeats, Team1]
[win_act, TeamClause1]	[Team1, loses]
[lose_act, TeamClause1]	[Team1, wins]
[win_act]	[win]
[lose_act]	[lose]
[[ObjectPrep, Team1], win_ps, [av, Team2]]	[Team2, defeats, Team1]
[win_ps, [av, Team2]]	[Team2, wins]
[win_ps, Team2]	[Team2, wins]
[win_ps, [av, Team1], [Team2]]	[Team1, defeats, Team2]
[[ObjectPrep, Team1], lose_ps, [av, Team2]]	[Team1, defeats, Team2]
[lose_ps, [av, Team2]]	[Team2, loses]
[lose_ps, Team2]	[Team2, loses]
[lose_ps, [av, Team1], [Team2]]	[Team2, defeats, Team1]

B Completing Facts

We have three different cases when we wish to complete the facts depending on how the fact looks:

1. The fact is `[Str]`, where `Str` is either win, lose or draw. The fact is completed to `[Subject, Str+s]` and processed again.
2. The fact is `[Team1, defeats, Team2]`. This fact is already complete. It is given priority 5 (highest priority) to reflect that the two teams were found in the same sentence together with a key.
3. The fact is `[Team1, Str]`, where Str is either wins, loses or draws. Such a fact is compared to all other sentences and completed depending on subjects and facts in these sentences. See table below for details. The final completion will be the one with highest priority.

The ”Fact” below is the processed fact. The ”Local Fact” is a fact occuring in the sentence that the fact is currently compared to. ”Subject” is the subject of the current sentence, ”Team” is a team belonging to the current sentence (with mark ”obj” if it is preceeded by an object preposition) and ”P” is the priority given to the completed fact. T1 and T2 are (different) teams.

Fact	Local Fact	Subject	Team	Complete Fact	P
[T1, draws]		T1	[obj T2]	[T1, draws, T2]	5
[T1, draws]		T1	T2	[T1, draws, T2]	3
[T1, wins]	[T1, defeats, T2]			[T1, wins, T2]	5
[T1, loses]	[T2, defeats, T1]			[T1, loses, T2]	5
[T1, wins]	[T2, loses]	T1		[T1, wins, T2]	3
[T1, loses]	[T2, wins]	T1		[T2, loses, T1]	3
[T1, wins]		T1	[obj T2]	[T1, wins, T2]	2
[T1, loses]		T1	[obj T2]	[T1, loses, T2]	2
[T1, wins]	[lose]	T2		[T1, wins, T2]	1
[T1, loses]	[win]	T2		[T1, loses, T2]	1

C A Small Dictionary from Swedish

Below is a dictionary of all Swedish terms used in this report in examples and explanations.

av	by
besegra	defeat (vb)
besegrades	was defeated
för	for
förlora	lose
förlust	loss, defeat (n)
mot	against
oavgjort	draw (*The game ended in a draw.*)
över	over
på	on
seger	victory
slogs	was defeated
till	to (prep)
vinna	win (vb)
vinnas	be won (*The game can be won.*)
över	over

Finally a translation of all Swedish sentences in the text:

IFK Göteborg lyckades till slut besegra ett svagt Örgryte i en spännande match. = IFK Göteborg finally managed to defeat a weak Örgryte in an exciting game.

Göteborg besegrade AIK. = Göteborg defeated AIK.

Göteborg besegrades av AIK. = Göteborg was defeated by AIK.

Matchen förlorades av AIK. = The game was lost by AIK.

IFK besegrades av AIK. = IFK was defeated by AIK.

IFK besegrade AIK som tidigare i veckan hade vunnit mot Örgryte. = IFK defeated AIK who earlier in the week had won against Örgryte.