# Detection of phone numbers, dates, time and names in handset text messages

**Camilla Kirkegaard**
University of Lund
Lund, Sweden
camilla.kirkegaard.282@student.lu.se

**Tobias Ek**
University of Lund
Lund, Sweden
tobias.ek.017@student.lu.se

## Abstract

To extract information from handset text messages is an area without a lot of previous research. This report contains an overview of some of the challenges. The areas we researched were gathering of corpus, witch is hard since it usually implies unrewarded contributions from handset users. With Java's regular expressions it was quite straightforward to reach 80% recognition of named entities.

Different tools can be helpful during program developing, for instance DTD provides a grammar check on the output and thereby gives early error detection. XSLT can be used as a way to visualize the result and make it easier during the development process since annotated corpus's in large quantities aren't particular readable for people.

It is necessary to develop a good way of measuring the performance to keep focus on the result when developing the program. Some other programs performance numbers are equally important to be able to evaluate the performance of the developed program.

## 1 Introduction

Handsets are all the time handling information, mostly either sending or receiving it, but the information is not often analysed or used by the handset applications. This is somewhat of an underutilised area and now as smart phones are being to more commonplaces there are lot research areas emerging.

What if the handsets started to display an intelligent behaviour and used the information handled. Just by analyzing the conversation between the two users, suggest to the user to schedule a meeting in his or her calendar or that the new number is detected that isn't in the users contact list perhaps it should be saved. This would provide freedom to the user, a freedom of not needing to keep things in her or his mind.

Sony Ericsson provided us with a description of this kind of an application that they want to be developed. They wanted first of all to see if there existed techniques good enough for this purpose. This project had as a goal to construct an initial framework for such application development. This also included doing an overview of the problems in the area and suggesting possible solutions.

A first step in such a research was to detect phone numbers, dates, time and names within text messages and annotate them properly. It also included constructing a method on how to evaluate the result and to compare it with existing standards. And finally should conclude with some conclusions about how usable the examined techniques are and suggest where further research is needed or which techniques we could involve.

## 2 Areas of research

### 2.1 Corpus

We started out with a corpus that was provided by Sony Ericsson. The corpus contains 214 text messages composed approximately from 2350 words. The corpus is in a JSON format and contains additional information such as caller id, service center, date, etcetera that was irrelevant for our purposes in this early stage and thus had to be ignored. We normalized corpus i.e. only lower case letters and divided the text into separate text messages.

The larger the corpus is the more likely it is that it is represent able and useful. A problem we faced was that many handset users did not want to voluntary give up their own private text messages for research possibilities. Having a too small corpus to work against increases the risk that the annotator will be overly adapted to that particular corpus and that it will be little use against other corpus's i.e. its over fitted. If we don't encounter an specific entity of something we want to annotate its likely we won't consider that entity at all. You might even have a sufficient large corpus but if the corpus is in Swedish and you try to use your annotator on a corpus in another language it will fail to reach the same performance. The major reason is that different standards are used in different countries for representing dates and times.

The small corpus and the possibility of an over fitted annotation program set the project on an unstable foundation. If the program is over fitted, it returns unfairly a high performance result against the current corpus but as the corpus grows larger it will degrade over time and against a different corpus it will perform poorly.

The issue with corpus gathering was not foreseen and the project was forced to continue even though the corpus was considered too small. We had anticipated incorrectly that our SMS corpus would grow larger during the project but a viable alternative could have been working on an email corpus until a sufficient large SMS corpus had been gathered.

## 2.2 Detection

To detect entities of name, phone numbers, dates and times was straight forward process and even with quite simple regular expressions a moderate performance was achieved. To detect phone numbers the following regular expressions were used to catch a variety of number formats:

[0][\\d][\\s-]?\\d[\\s-]?([\\s]?\\d){6,7}
E.g. 046-123456.

([00]|[+])\\d{2}[-\\s]?([\\s]?\\d){6,}
E.g. +4646-123456

To detect dates we developed this regular expression that is naive but still frames the majority of dates:

\\d{4}|\\d{2})[.-/]?[01]\\d [.-/]?[0123]\\d[^\\d]
e.g. 2010-01-11 and 10.01.11

With times we had to deal with the presence of "am" or "pm" and the different denominator use of either '.' or ':' which we did in the following regular expressions:

(\\D)([012]?\\d[:.][0-5]\\d(am|pm)  e.g. 11:25pm

[012]?\\d[:.][0-5]\\d     e.g. 23.25

[012]?\\d(am|pm))    e.g. 11pm

Since we are aware of the possibility of over fitting our regular expressions against the small corpus, we decided to not put much effort on develop the expressions much further at this point than what is shown in previous examples. Likewise if we had introduced our own imagined entities in attempt to expand the small corpus we would likely encounter the very same over fitting issue. We redirected our focus to spend the time on getting an orientation of the different problem areas and look at viable solutions to accomplish the project goals.

Some conceptual problems were encountered as how to interpret a number like "091011", is it 9th of September 2009? Numbers like 1234 might be 12:34 o'clock, a direct number under a company's telephone switch or something completely uninteresting for annotation.
Handset user don't always conform to any standardised way to represent time, date or phone numbers they are very likely to try to compress a sentence to as few characters as possible. Numeral examples like 31.e which easily is deduced from the context to mean 31th of some month.

## 2.3 Standard annotation

It is important to comply to a well used standard since it encourages and simplifies further development if ever reused. By this reason we choose to comply with the Message Understanding Conferences (MUC) annotation scheme as its one of most popular one. MUC includes annotation to handle a wide range of different entities in every conceivable permutation out of these we use TIMEX and ENAMEX.

TIMEX3 covers times and dates and is a very comprehensive standard but at this point we have no need for it's full complexity. We use only the following two formats.

<TIMEX3 type="DATE" value="XXXX-10-20">oktober 20</TIMEX3>
<TIMEX3 type="TIME" value="T15:45">15.45</TIMEX3>

Since there are different ways of representing times and dates, the value attribute has to be normalized for further use just as for phone numbers. For instance a partial date as "October 20" XXXX is a filled as place holder for the year to indicate the information is missing, the TIMEX scheme we follow represent dates in the format "YYYY-MM-DD".

ENAMEX is used to annotate several different named entities as organizations, persons and locations. We considered only the person identification in this project even though we had location in mind for later use in our master thesis.

<ENAMEX type="PERSON">jens</ENAMEX>

To annotate phone numbers we created our own annotation since there was no common standard scheme in use. Our phone number annotation uses the same format as the others and is in a normalized format in the attribute value where they are stripped of dashes, spaces and parenthesis.

<PHONENUMBER value="046123456">046-123456</PHONENUMBER>

## 2.4 Gold annotation

To be able to measure the annotator's performance for the entities, it is necessary to have a gold standard which is a hand annotated part of the corpus. But in our case the gold standard had to be whole corpus. A gold standard annotation means that it is annotated by a human and thereby should be an ideal annotation.

To gold annotate a part of the corpus is not always trivial, it raises several questions and things have to be thought through before you start the process. For instance should there be an duration attribute attached to TIMEX annotation if so when should it apply if there are two dates that are close together or maybe just if there is an obvious denominator between them.

The corpus includes some tricky entities that we as human beings can easily detect especially with help of the context but isn't realistic found by a (simple) automated annotator. We ended up decided to be strict in our annotation and used the rule "if we understood something as date we annotated it as such". This could imply that entities like "let's meet on the 25" should be annotated as the 25th in either this or the coming month, depending on the current date. These kinds of entities are not detected by our annotator yet and played a part of why it was hard to increase the performance of our annotating program after a certain level of success.

To hand annotate a corpus is a very time consuming process and we also had to redo the annotation a few times, since our annotation scheme and our understanding on the proper way to annotate entities changed during the project.

## 2.5 DTD as a grammar check

DTD provides a grammar check for the annotated output. It enforces that the annotations are constructed in a standard way and are well formed. Since DTD detects malformed annotations its can be very helpful during developing of the software to detect and locate errors quickly.

We used a fast test cycle combined with the DTD grammar check. This combination gave a good clues of which commit that went wrong and speeds up the tracking of newly introduced bugs. It gives a fast and repeatable test that everything that worked before a change was still working after the change.

## 2.6 Visual overview with XSLT

Our XSLT program takes an XML file as input and transforms the XML into XHTML. Viewed in a web browser the XHTML provides a visual overview of how the annotation has been carried out. As our XHTML highlights each type of entities of dates, times, and phone numbers in its own colour. It made it relative quick and easy to compare against the gold annotated corpus if the annotations are correct, missed, partial or incorrectly annotated.

## 2.7 Evaluation

Both the DTD and XSLT were an intermediate step to measure performance but they didn't provide exact results. To evaluate the result it is required to gold annotate a slice of the original corpus and then compare it with the output that the annotation program provides. The standard is to use a harmonic mean, generally called F-measure, to describe the performance. F-measure is calculated as a mean value between the precision and the recall.

$$Recall=\frac{retrevied\ tags \cap relevant\ tags}{relevant\ tags}$$

$$Precision=\frac{retrived\ tags \cap relevant\ tags}{retrived\ tags}$$

$$F\ measure=\frac{2 * Precision * Recall}{Precision + Recall}$$

Precision is "how many of the entities are accurate" and recall is "how many of the entities were found". We found it convenient to both evaluate how good the overall performance was, as well as to evaluate how good each entity type performed. This gave a fast clue of how well a specific feature of the program worked. A good evaluation program is necessary to be able to develop in a result focused way.

## 2.8 Comparison

As soon as the annotating program together with the evaluation program starting produce outputs, performance started to climb and the most obvious errors could be corrected. Next step was to compare it with existing software on the market to get a hint of real performance. We used the Macintosh's Apple Mail that is included in OS X v10.6 which is its default mail program it's a very capable program and truly is the state of the art.

The initial results were too optimistic from our point of view but as it turned out, it was not comparable, since our annotator also annotated names from a contact list and some Swedish words that the Apple mail can't handle. This made the comparison unfair and by this reason, we cut our annotators ability to recognize names and Swedish words to have them compete on equal grounds. After the revised comparison our annotator and Apple Mail performed with quite similar result. Still there were some key differences, our annotator has slightly more entities annotated than the Apple Mail but it still outperforms our annotator because it has a higher precision on its annotations.

As can be seen on figure 1, there are just 91 entities (excluding names) from the 214 text messages reveals that there is a high risk that our program is over fitted which puts a shadow on the result. Then add the knowledge that the Apple Mail is being more language neutral than ours and thereby has to consider several more conventions of writing date, times and phone numbers, as generality usually comes with a cost and in this instance it affects overall performance. To evaluate the Apple Mail we simply email our self the XML file and then manually checked if the Apple Mail was able to detect each entity, obviously a very crude method and the realization of an automatic comparison technique dawned on us.

Our annotator

| | Gold | Correct | Incorrect | R | P | F-measure |
|---|---|---|---|---|---|---|
| Name | 51 | 51 | 0 | 1 | 1 | 1 |
| Date | 39 | 28 | 5 | 0.72 | 0.85 | 0.78 |
| Time | 41 | 34 | 0 | 0.83 | 1 | 0.91 |
| Phone | 11 | 7 | 1 | 0.64 | 0.88 | 0.74 |
| Total | 142 | 120 | 6 | 0.85 | 0.95 | 0.9 |

Our annotator, names are excluded

| | Gold | Correct | Incorrect | R | P | F-measure |
|---|---|---|---|---|---|---|
| Date | 20 | 12 | 8 | 0.6 | 0.6 | 0.6 |
| Time | 41 | 34 | 0 | 0.83 | 1 | 0.91 |
| Phone | 11 | 7 | 1 | 0.64 | 0.88 | 0.74 |
| Total | 72 | 53 | 9 | 0.74 | 0.85 | 0.79 |

Apple Mail(OS X v10.6)

| | Gold | Correct | Incorrect | R | P | F-measure |
|---|---|---|---|---|---|---|
| Date | 20 | 15 | 1 | 0.75 | 0.94 | 0.83 |
| Time | 41 | 28 | 0 | 0.68 | 1 | 0.81 |
| Phone | 11 | 7 | 1 | 0.64 | 0.88 | 0.74 |
| Total | 72 | 50 | 2 | 0.69 | 0.96 | 0.81 |

# 3 Summary

## 3.1 Gathering a corpus

To gather a corpus of handset text messages is harder than it might seem, the average handset users aren't willing or doesn't or see why they should contribute with their text messages to further research. Some kind of reward system might be a solution to address this matter. The need of a corpus of at least 10,000 words and preferably much more is required to overcome the over fitting problem.

### 3.2 Detection

It is quite straight forward to detect the majority of the entities in the corpus with relatively simple regular expressions, in our case a detection rate near 80%. To refine the numbers more sophisticated regular expressions and probably a combination with other techniques are needed. The area of machine learning might be of interest.

### 3.3 Annotation

MUC provides a comprehensive standard for annotation of dates, times, names and locations, where the latter will be useful in a further application development. For phone numbers there is no widely used standard yet and we have to develop our own.

### 3.4 DTD

A DTD showed itself to be useful in grammar checking the result after any change to the annotation process. The DTD provided instant errors if the output did not corresponded to the rules we had in place. The DTD is mostly useful in the early stages of development but is convenient later on as well.

### 3.5 XSLT

It is hard to read annotated text with annotations. It is hard to see if any mistakes were made, as in wrong, lacking or double annotation. The XSLT provided an XML file which was viewable within a web browser and each kind of entities, e.g. phone number, was easy distinguished with its specific colour.

### 3.6 Evaluation

It is standard to use a harmonic mean, F-measure, to describe the performance. A good evaluation program is essential to be able to develop in a result focused way.

### 3.7 Comparison

When having a program and evaluation program that works the final part has to be to compare the performance with existing programs to get an idea of how good it is and which parts that has to be improved. This part would benefit greatly by an automated process to avoid unnecessary manual work.

### 3.8 Future work

To reach a higher score for recognised entities, there are different things that could be done. Improving the regular expressions, make them more exhaustive to cover more kinds of entities, more work is also needed in improving their accuracy on matches. A convenient way of adding more regular expressions for specific matches is needed without having to increase the complexity of a single expression. Keeping the expressions simple would have huge benefits in debugging and improve matching. We also ran into a undocumented bug in the Java regular expression engine that hinder development. There might be libraries that are more capable or more up to specification that could be used instead.

Taking context into account seems to be an important next step for the application. Context could involve certain key words that might imply that a meeting is about to occur between users. We would like to be able to match occurrences like "I'll see you at one". Knowing that "at" is a key word if followed by a number, others keywords could be o'clock, meet or see.

We would like to give the program an appearance of an intelligent behaviour for instance to analyse an chain of messages between users, saving the annotations found in previous messages and see if the receiver answers back with information that has been annotated.

Example given:

**A** Sends a SMS including a date.
**B** Returns the SMS including a time and a location.
**A** Receives the SMS.

Here we would like to suggest to person A to schedule a meeting with B at the given location, date and time. If then the conversation continues:

**A** Sends an accepting phrase, e.g. "ok".
**B** Receive the SMS from A.

Here user B would be prompted to schedule the same meeting.

Utilise the meta data that comes with the corpus to aid in filling partial dates, connecting names to numbers, see the time frame messages are being sent if they are a candidate for scheduling a meeting.

We would like to examine the possibilities of using machine learning techniques and explore ways of combination them with regular expressions.

**Acknowledgments**
We want to thank Pierre Nugues for a good guidance and support throughout the entire project. Håkan Jonsson at Sony Ericsson for assisting us with the project idea and a corpus. Lund's University for lending us a Mac and providing technical support with it.

## References

Pierre Nugues. *An Introduction to Language Processing with Perl and Prolog*. http://www.cs.lth.se/home/Pierre_Nugues/ilppp/

ENAMEX http://cs.nyu.edu/cs/faculty/grishman/NEtask20.book_9.html#HEADING30

TIMEX *Guidelines for Temporal Expression Annotation for English for TempEval 2010*, www.timeml.org/tempeval2/tempeval2-trial/guidelines/timex3guidelines-072009.pdf.