Text ⇒ TPEG
Automatic generation of language independent traffic messages

Fredrik Bergholtz and Florent Delgoulet

4th October 2002
Abstract

This document describes the problems of converting written Swedish text into language independent traffic messages and suggests an approach for solving them.
## Contents

1. **TPEG - Introduction** ............................................. 1
   1.1 Road Traffic Messages ............................................. 1
   1.2 TPEG tables .................................................... 2

2. **tpegML - a formal language** ..................................... 2

3. **Traffic incident management system—oJJe** ......................... 4
   3.1 Location references .............................................. 4
   3.2 Incident reports .................................................. 4

4. **Manual generation** .......................................................... 4

5. **Requirements for automatic generation** .......................... 5
   5.1 Tagged corpus ...................................................... 5
      5.1.1 Purchase tagged corpus ...................................... 6
      5.1.2 Construct tagged corpus ..................................... 6
      5.1.3 Tag set? ...................................................... 7
   5.2 Parsing .............................................................. 7

6. **TPEG⇒Text** ................................................................ 8
1 TPEG - Introduction

TPEG, the acronym stands for “Traffic Protocol Experts Group” and it is the name of a system specification for delivering digital, language independent traffic messages over most digital bearers. Like the names JPEG and MPEG, TPEG also identifies the group of people involved in creating the specification.

The general idea of TPEG is that a stream of encoded bytes is sent, and continuously repeated over a digital bearer. The stream is divided into several services. A service is built up by several service components containing the actual messages. Services can be used to specialize the content for different users, one service could provide information on a national level, another could be focused on regional messages. TPEG also include mechanisms for conditional access (CA) which allows a service to be specialized for paying customers.

There is an expansive definition of the service components of TPEG, currently two applications are defined and a third is under construction. An application defines the content of a service component. The applications are Service Network Information (SNI), Road Traffic Messages (RTM) and Public Transport Information (PTI). SNI describes the contents of the service, this means that it indicates the application type and other information about each service component. RTM contains actual traffic messages including reports on accidents, congestion, road conditions and weather, to name a few.

1.1 Road Traffic Messages

TPEG contains several different applications, but this report will concentrate on the road traffic message application. But we hope that the methods covered here will be applicable to other applications as well.

A road traffic message is a hierarchical structure, defining more and more details on each level. The top level defines what type of message it is, accidents, obstructions, activities, road conditions among others. Each of these top level components has a different sub structure. For example an accident can have the following message components on the next level: position, animals, vehicles and people.

**position** describes the position of the accident relative to the road, values are like “driving lane 1”, “emergency lane” or “roadside bank”.

**animals** describes any animals involved in the accident. The animals component have a sub structure defining the position of the animals, the problem with the animal and information about the animals. Information of the animals contain information about the animal type, as in species, and the size of the animals.

---

1Note the difference between a service component and a message component.
vehicles describes any vehicles involved in the accident. This component also have a sub structure defining more details.

people describes any people involved in the accident. This component also have a sub structure defining more details.

Many of the sub components are re-used in the components, especially the position component that is present in almost every message component. This means that position information can be given in the exact same manner for each of the specific information regarding this accident.

1.2 TPEG tables

Almost all values of the message components in a TPEG message is defined as an index in a fixed table of words. For example the values mentioned in the previous section are referenced by their table and index value:

<table>
<thead>
<tr>
<th>Value</th>
<th>Table</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>driving lane 1</td>
<td>rtm10</td>
<td>1</td>
</tr>
<tr>
<td>emergency lane 1</td>
<td>rtm10</td>
<td>44</td>
</tr>
<tr>
<td>roadside bank</td>
<td>rtm10</td>
<td>60</td>
</tr>
</tbody>
</table>

These tables are what enables the language independence of TPEG. The semantics of each value in each table is decided, but the text that is to be presented in the receivers is not. This means that, as long as the agreed semantics match, the receiver manufacturers are able to create a text in any language. The simplest method for generating text from TPEG could be to use different tables for each language where the values are simply replaced by values of the language in question. This method will of course not be able to generate grammatically correct text without lots of ad hoc rules. In section 6 we will discuss how to make better text out of the TPEG messages.

2 tpegML - a formal language

While the TPEG native data format is an encoded binary stream, the TPEG project has come to the conclusion that we need a more easily manipulated data format in the production phase of the TPEG messages. Therefore tpegML—an XML implementation of TPEG, was conceived. The hierarchical nature of XML fits perfectly to TPEG. Instead of immediately encoding the messages to a binary byte stream in the message generating software, we use standard XML tools to generate tpegML that in the final phase of stream generation will be encoded for the stream.

An example message: There has been an accident involving 50 cars at the intersection of A12 and A128 near Brentwood, Essex. The accident was caused by
thick fog, restricting visibility to 20 meters. The northbound lanes are closed.

Expressed in tpegML, this message looks like:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE tpeg_document PUBLIC "-//EBU/tpegML/EN" "tpegML_05.dtd">
<tpcg_document>
  <road_traffic_message message_id="123"
    version_number="1"
    message_generation_time="2002-04-03T13:03:00Z"
    severity_factor="&rtm31_4;">
    <!-- Location is on A12 in Brentford, Essex -->
    <tpcg_loc_container language="&loc41_30;">
      <location_coordinates location_type="&loc1_5;">
        <WGS84 longitude="-0.1337" latitude="51.52641"/>
        <descriptor descriptor_type="&loc3_7;" descriptor="A12"/>
        <descriptor descriptor_type="&loc3_8;" descriptor="A128"/>
        <descriptor descriptor_type="&loc3_24;" descriptor="Brentwood"/>
        <descriptor descriptor_type="&loc3_25;" descriptor="Essex"/>
      </location_coordinates>
    </tpcg_loc_container>
    <!-- Accident in thick fog involving 50 vehicles -->
    <accidents number_of="1">
      <position position="&rtm10_37;"/>
      <vehicles number_of="50">
        <vehicle_problem vehicle_problem="&rtm3_22;"/>
      </vehicles>
    </accidents>
    <visibility>
      <obscurity obscurity_problem="&rtm17_2;"
        visibility_distance="20"/>
    </visibility>
    <network_conditions>
      <position position="&rtm10_37;"/>
      <restriction restriction="&rtm49_1;"/>
    </network_conditions>
  </road_traffic_message>
</tpcg_document>
```

The values of the tables are naturally transformed into XML entities, also the
document type definition (DTD) of tpegML is divided so that all of these enti-
ties are defined in separate files. This means that the simplest way to generate language specific text could be performed by simply using different files for defining the entities. An example of how this could be used, in conjunction with XSL stylesheets can be seen at a BBC website: http://www.bbc.co.uk/travelnews/xml/

3 Traffic incident management system—oJJe

At Swedish Radio (SR), a traffic incident management system is used by all traffic reporters. This system is named oJJe, and is especially developed for SR. All regional radio stations can plug in to this system to see what is happening all over Sweden. All incident reports are immediately visible for every traffic reporter. Another feature of this system is that external information can be entered to the system in many ways. This enables, for example the police, the road authorities or various public transport operators to enter information that will be directly visible for the traffic reporters. The traffic reporter generally decides if an incident is worth publishing, and if so he or she will usually re-write the incident in a shorter version, one that is adapted for being published either as text on the SR web pages or to be read in the radio broadcast.

3.1 Location references

The oJJe system is divided into several parts, one of these parts is a map system. Here the traffic reporters can see the location of all incidents. And for creating new incident reports the map can be used to pinpoint the location.

Currently an effort is being made to enhance the map functions to automatically generate the TPEG location reference containers. These containers contain both coordinate location and textual references to describe a location. Textual references include items such as city names and various combinations of road names to uniquely defined locations.

3.2 Incident reports

The incident reports are defined by a category, a title and a text. In addition to these other attributes exist, but most of these are only used as raw input for generating those three. Of course, each incident can be linked to a location defined in the map system.

4 Manual generation

Currently traffic reporters manually generate TPEG messages by selecting TPEG components from several drop down boxes and lists. This approach takes a lot
of time from the reporter, and it is not intuitive. The reporters must know a lot about the details of TPEG to use it, and this should not be part of their jobs.

5 Requirements for automatic generation

First of all, we should remind you of the requirements of the solution to be held:

- it should exactly represent and be faithful with the author’s mind. He writes himself the original message, then this message is processed and broadcast to the radio listeners. When the radio receivers present the message the presentation can be in a different language. All through these operations, the message validity should be kept, so no bad interpretation should be made at all, or the sense of transferring the message would be lost.

- its processing needs to be fast, because a piece of information that is not given in time may have no value at all: the operations affecting the original message should then be done nearly live (no too long processing).

- it must be syntactically and semantically correct, that is, understandable by everyone.

Unfortunately all these requirements look too ambitious, but that is the ideal, i.e. the goal, we want to reach. Since we have no precise idea on how to do it yet, please note that all the following statements are only suggestions and questions. They will not be fully answered in this report.

Note: Some of the suggestions below may require that the original messages do not contain any spelling mistakes because it would be very difficult to generate a new text from an incorrect one. Of course, in real life, everybody can do mistakes and some of them may really exist since the messages will be quickly written; so a spell checker must undoubtedly be involved.

As we already said above, to be distributed, the messages should not be saved in their original form, but in a standard common form, completely independent of the original language.

5.1 Tagged corpus

What is the aim of a tagged corpus? To parse the original message in order to isolate and understand each word contained in it: that is to identify its kind, if possible its stem, its current form and its link with the rest of the message.
5.1.1 Purchase tagged corpus

Existing tagged corpuses are usually large, they may list all the words that the reporters might use in their messages, and even much more than those. The language generated by such corpuses is usually wide, and they make the inventory of words taken from many different contexts. And they might be very reliable.

Is it very useful to have a tool that can recognize unused words in the traffic information domain? Is it possible to use one of these corpuses since the language manipulated would not be as large as the one that can be processed by it?

Moreover, since the corpus would contain many unused words, would not it take too much time to parse the messages? And would the results be as good as our requirements?

Lastly, would this solution be profitable since the beginning investment is likely to be high?

5.1.2 Construct tagged corpus

The solution that looks the most interesting for us is to build our own corpus: it would then exactly fit to our needs, but it might be difficult to develop and implement it, because our needs may change quite often (the reporters do not use the same vocabulary in all their messages and currently the TPEG specifications are not fixed).

Two possibilities cross our minds:

- First, it is possible to manually tag a large number of existing messages taken from the archive of the incident management system, for instance with statistics on more used typical messages as a fundament, and then use the tags back to tag the messages to come.

  Problems encountered: such a solution would take a lot of time to be settled, and would make it difficult for the system to tag never before used traffic messages, because it would not know them. Moreover, are we certain that there are any common patterns at all in the messages (that is, can the older messages help us in the future?), since they might be written by different reporters at different times about different subjects?

  Possible solutions: the inventory is a full-time job for several days/months but it is not insurmountable. Once the inventory work done, it would be possible to assign a person to check continuously that there is no problem at all in the tagging operation: for instance, if a message can not be processed, we can imagine that it is because it was impossible to tag it, so because the message components did not look like existing ones. A piece of information could then be sent to the “checking” person, so that he or she could clear things. But this would require a lot of permanent
attention from him or her, and a lot of money too because this employee
would be specialized (knowledge enough to be proficient) and his only job
would be the checking.

Accidentally, there is a kind of paradox with the previous solution (see
§5.1.1): we thought that purchasing a tagged corpus would lead us to
a too wide language not fitted to the reporter’s language, and now this
solution is likely to lead to a too narrowed language. So one possible
solution might be to mix all this...

- We could also try to tag the traffic messages automatically using an ex-
isting system like “Granska”, which seems to be efficient and reliable to
parse the messages correctly.

Problems encountered: although, we can not always be sure of it, and we
would then once again need a person to correct manually the possible bad
results. Then, are we 100% sure that the corpus is large enough to tag all
the possible messages?

5.1.3 Tag set?

Either we could base the generation on the Swedish grammar, or on our own
grammar.

• Swedish grammar? This solution would be quite eloquent and it may
make things easier when a problem occurs, because the solving might be
simpler. Moreover, Swedish grammar is fixed and exhaustive, while our
own grammars may not take care of all the possible cases and may be
modified with time. But what if the original message is not written in
Swedish? That may be an extension of the project...

• tpegML grammar? tpegML is precisely the aim we try to reach. The work
would be done once the messages are translated into tpegML. It would be
easy to export the results. However, there are some limits that we may
encounter: how can we be sure that every message can be translated into
this grammar?

5.2 Parsing

As a first step we tag the words according to word classes and/or TPEG table
indices. We could mark the words in a message with word classes as well as with
the possible TPEG table indices. For instance, if we go back to the example
given above we show how a part of a message can be tagged:

The northbound lanes are closed
(The)det (northbound lanes)rtm10_37 (are)vb pres (closed)rtm49_1
If a word does not exist in a TPEG table it can be tagged as "not in TPEG table", this can be used to fill unknown components by the syntax parser. The syntax parser could continue to tag this entire phrase as:

\[
\left( ((\text{The})_{\text{det}} \text{(northbound lanes)}_{\text{rtm10.37}})_{\text{position}} \right)_{\text{network}} \left( (\text{are})_{\text{vb}} \text{pres} (\text{closed})_{\text{rtm49.1}} \text{restriction} \right)_{\text{conditions}}
\]

After this tagging, we know that this message is syntactically correct and really do represent a valid TPEG message. Generation of tpegML is an easy task, using the tag information.

6 TPEG⇒Text

An interesting task is to analyze what happens at the other end of the TPEG transmission. Currently receiver manufacturers concentrate on presenting TPEG information graphically, as an icon in the map of a navigation system in a car. Future developments will include text presentations, and even speech synthesis. Since the main application of TPEG is road traffic messages, the receivers will mainly be used in cars. And this makes speech presentation more important. Another interesting aspect of the text presentation is the generation to different languages from the same source.