Summer School on Natural Language Generation, Summarisation, and Dialogue Systems

Open Lab Sessions

Yaji Sripada

Welcome to the open lab sessions! During this week, you will learn to build a simple NLG application hands-on. Building an NLG application, like building any other software involves the standard stages of requirements analysis, software design, coding and testing. We recommend you read Reiter and Dale (1997) paper on 'Building Applied Natural Language Generation Systems' (http://homepages.abdn.ac.uk/e.reiter/pages/papers/jnle97.pdf) to learn the most commonly used methodology of NLG software development. You are of course strongly recommended to read the book Reiter and Dale (2000) 'Building NLG Systems'. In the case of NLG, all the software development stages are heavily informed by corpus analysis. This means, you need to analyse examples of human written texts to understand the requirements for your NLG application. Also you need to analyse corpus to acquire knowledge of the sublanguage and the information content communicated in the texts to design an NLG application.

Your Main Task

A text corpus contains human written texts belonging to a specific domain. For these open lab sessions we selected the domain of SCUBA (Self Contained Underwater Breathing Apparatus) diving. We give you a corpus of human written descriptions of SCUBA dive profile data (you will learn what they are soon) and your task is to build an NLG application to auto-generate descriptions similar to the corpus examples. This means you need

- To specify the requirements of YOUR NLG APPLICATION
- Design YOUR SOLUTIONS to the various subtasks such as document structuring, lexicalization, aggregation and realization.
- Write YOUR Java code to implement your design
- Make a demo of YOUR NLG APPLICATION (Testing your system!!)

Building a complete NLG system end-to-end can be quite a laborious activity. Therefore we give you a couple of Java libraries to reduce your workload:

1. ScubaNLG is a Java library to read, analyse and interpret scuba dive profile data downloaded from a dive computer. This library does all the plumbing work you need so that you focus on the NLG part of system design and development.
2. SimpleNLG (https://code.google.com/p/simplenlg/) is an off-the-shelf realizer for English. This library helps you to generate grammatically (morphologically and syntactically) well-formed English sentences.

Overall Plan

Day	Description
Monday	Introductions and team building, installing and running ScubaNLG and SimpleNLG, group discussions about requirements and design
Tuesday	Starting application development-focusing on using simpleNLG to produce sentences similar to corpus from the information available from ScubaNLG
Wednesday	Continuing to work on sentence level generation subtasks
Thursday	Document structuring
Friday	Demo

On each day, teams work on a specific set of tasks. Each task is achieved following the sequence of steps below:

1. Task definition and understanding –teams discuss and understand the task
2. Solution Design – teams work out their solution to the task
3. Coding – teams write the required code
4. Group discussion – at the end of the task all the teams come together to discuss the task

We believe that the Open Lab sessions will deliver maximum benefits when the group discussions in step 4 involve everyone!

Tasks

Task	Description
Task 1	Introduction and Team Building
Task 2	Install and run the ScubaNLG application
Task 3	Study the ScubaNLG application
Task 4	Corpus Analysis. There are ten example texts and their corresponding dive profile graphs and feature vectors. Thus there are ten subtasks for analysing each of these corpus texts.
Task 5	Architecture level discussion about requirements and design
Task 6	Planning simple sentences that have information readily available from ScubaNLG. Coding to auto-generate each of these sentences is a separate subtask.
Task 7	Planning complex sentences that require additional planning. Again coding to auto-generate each of these sentences is a separate subtask.
Task 8	Document structuring using simple ideas first. Advanced ideas to be discussed as a group.
Task 9	Making your system ready for the demo. You may want to make your application generate texts better than corpus texts!!!

New to Eclipse IDE?

We will use Eclipse IDE for Java during these open labs. If you are new to it, and are willing to learn, you find a lot of online help such as http://eclipsetutorial.sourceforge.net/Total_Beginner_Companion_Document.pdf. Please use any online resource you like (including You Tube videos). Please do not proceed to the development tasks without learning to use Eclipse.

Task 2 - Install and run the ScubaNLG application

1. Download the ScubaNLG.zip file into your Eclipse workspace folder
2. Unzip the file which creates ScubaNLG folder in your workspace folder
3. Start your eclipse and click File->Import
4. In the Import Dialogbox, select General->Existing Projects into Workspace and click on next
5. Check Select root directory and browse to your Eclipse workspace folder
6. Select the ScubaNLG root folder
7. Click on Finish
8. Add simplenlg-v4.4.2.jar to your project's referenced libraries
9. Open the class ScubaNLGMain.java for editing
10. Select one of the MYJDBC calls in the constructor based on yours database (MSAccess or MySQL)
11. If you choose MySQL then add mysql-connector-java-5.1.36-bin.jar to your project's referenced libraries. Add your user name and password to your MYSQL server details.
12. Now run the class ScubaNLGMain
13. A map view of the dive locations is displayed where each dive location is marked with a red dot labelled with its dive number.
14. Click on the red dot to display the dive report for the selected dive.
15. The dive report has two parts: Graph and Text
16. Text report again has two parts: Corpus Text and Computer Text.
17. Computer text is currently the feature vector computed by ScubaNLG and your task is to write NLG module to ScubaNLG to auto-generate textual report similar to the corpus text using the information from the feature vector.

Recreational Scuba Diving in a paragraph (rather long)

It is a popular sport where divers carry their breathing cylinders. Typically these cylinders contain enriched air (increased proportion of Oxygen and reduced proportion of Nitrogen than normal air). Recreational divers receive training and get certified to dive. Obviously a large part of this training involves training to use the diving kit, particularly the breathing cylinder. An important part of modern diving kit is a dive computer which looks more like a wrist watch. This computer is capable of measuring the depth, time, water temperature etc of a dive and use this data to compute the amount of gases absorbed by the different tissues of the body. Softer tissues absorb at a quicker rate while harder ones do it at a slower rate. Our body tissues absorb gases even without going for a dive. But normally atmospheric pressure (~1 bar) determines the amount of gases our body absorbs. Under water there is this additional pressure of the water column (depending on the depth of your dive) making your body absorb more gases. The deeper you dive the more the absorbed gases. Human body gets rid of these absorbed gases naturally as the diver ascends to the surface. However the speed of ascent is an important parameter. If the diver ascends to the surface rapidly, the absorbed gases form micro-bubbles in the body, very similar to how a fizzy drink bubbles out when the cork is opened. These micro-bubbles travel through the body and collect at joints. This causes what is known as decompression illness (DCI) or bends. As a Scuba diver you like to avoid DCI decisively. This means, you need to keep an eye on the depth, bottom-time (time you stayed in the dive before you start to ascend) and ascent speed as these are the three important features of a dive. A dive computer records the complete path taken by the diver underwater which is known as the dive profile. ScubaNLG reads dive profile data and computes the depth, bottom-time and ascent speed of a dive. Read '*RDP InsforUseMet.pdf*' document in the ScubaNLG folder for additional details.

Task 3 – Study the ScubaNLG Application

The application has four packages, *hci*, *analytics*, *nlg* and *app*. You will work mainly on adding code to the *nlg* package. In the *hci* package you may want to inspect mainly *AlladinLogMapView.java* class. This class implements a mouse listener and when mouse is clicked on the red dots in the map view the associated dive profile is read, analysed and reported on. In the *analytics* package you may want to explore *DiveFeatures.java* and *DiveletFeatures.java*. In the *nlg* package, the class *ScubaReporter.java* implements the Reporter interface. You are expected to add NLG functionality to *ScubaRepoter.java*.

Task 4 – Corpus Analysis

The ScubaNLG application displays a corpus text for each dive. Your job is mainly to understand the sublanguage of these texts and the content they communicate.

A sublanguage of a natural language (such as English) is made up of words, phrases, clauses and discourse structures that are subsets of that natural language and is used in a specific domain. For example, English weather reports use a small subset of words, phrases and clauses in the English language. Thus English '*weatherese*' is the sublanguage of English weather reports. What is sublanguage of scuba reports? Is it simple or does it involve complex linguistic constructs? What are the more frequent phrases (e.g. noun phrases) in the corpus texts? What is their structure? Answers to these questions will be useful in scoping your development work. You will return to corpus analysis as part of the other tasks later. For now though it is useful to identify the different sentence and phrase types your application should auto-generate.

The next step in corpus analysis is to understand the content of the corpus texts. The ScubaNLG application provides input to this content. Please check if all the content you need to auto-generate the scuba reports is computed by the ScubaNLG application. Note down the gaps. (See section 3 in Reiter and Dale 1997 for details on corpus analysis for NLG).

Task 5 – Architectural Level Discussion

Most NLG applications use the famous pipeline model (Reiter and Dale 2000). It is worth spending a few minutes discussing if this is a good fit for generating scuba reports.

Task 6 – Planning Simple Sentences

All the previous 5 tasks are preparatory steps for the system development. From here on, it is all about actually building the application.

New to SimpleNLG?

Please do not proceed with your system development if you need to learn SimpleNLG. A good learning resource is https://code.google.com/p/simplenlg/wiki/Tutorial.

SimpleNLG library is already added to your project. So you can go through the tutorial by adding code to the *ScubaReporter.java* class. You may want to try sentences from your corpus in addition to the tutorial exercises. An important subtask of planning simple sentences is lexicalization (section 4.1.4 in Reiter and Dale 1997).

After you learn to use SimpleNLG, start with a list of simple sentences your system should generate and design your scheme to generate them using SimpleNLG. Can you design your code in such a way that same code can be used to produce several different sentences? Please inspect the enum *AscentType.java* in the *nlg* package.This is given as an example which you may not want to use. This code allows you map ranges of ascent speed values to appropriate adjective. So this enum is a generic lexicalizer for expressing ascent speed using an adjective.

Task 7 – Planning Complex Sentences

Some of the corpus sentences may require more functionality than what SimpleNLG offers. For example texts that describe two successive dives involve *referring expression generation* (a very simple case, of course). A lot of NLG research is devoted to the topic of referring expression generation (section 4.1.5 in Reiter and Dale 1997). Similarly *aggregation* is another research topic in NLG (section 4.1.3 in Reiter and Dale 1997). By the end of this task you are expected to have created functionality for generating all the individual sentences in your corpus. You may also like to work on generating sentences using different words, phrases and clauses compared to the corpus sentences. Please bear in mind that your main objective of this system building exercise is to learn to build NLG applications.

Task 8 – Document Structuring

Even a causal glance through your corpus shows that most texts are multi-sentential. Document structuring to auto-generate multi-sentential text is an exciting research topic (https://en.wikipedia.org/wiki/Document_structuring). In most practical applications a schema based approach is adequate. Please check if you could use the same approach for your application.

Task 9 – Preparing the demo

Think of exciting features you could add to the application, your chance to show-off in the demo!!!

That’s you after the *nine tasks to NLG*!