

Blather

A language-learner and sentence generator

Index

1. Authors
2. What is Blather? Basic concepts
3. Implementation
4. Examples
5. Interface (also contains usage instructions)

1. Authors

The Blather project was created by Katie Collins and Eric Goodwin, for their fall 2007 CS 1332 project, in November and December, 2007. Questions may be directed by e-mail to katie.collins@gatech.edu and eric.goodwin@gatech.edu.

2. What is Blather? Basic concepts

The inspiration for this project was the MegaHAL project (<http://megahal.alioth.debian.org/>), which reads in data from users and attempts to form a coherent idea of sentence structure and vocabulary from the data it reads. It then attempts to generate responses based on what it has learned. Blather's goals are the same, but the way in which it accomplishes it is very different.

Blather deals solely in sentences. Based on data it reads in from the user, either in the form of individual sentences or text files containing sentences, it forms an adjacency list graph of the words, with each word forming a single node. Words contain links to other words, as well as information about how many times they have appeared and how likely they are to be at the beginning and end of sentences. Generated sentences are created randomly, but the links between words are weighted, which ensures that the

most frequent word connection is more likely to be chosen (see the graph diagrams below for more information).

After data is read in and the graph is formed, sentences are generated based on a set of keywords that act as starting points in the graph. Of the keywords it finds in the graph, it picks the one most likely to be at the beginning of a sentence, and uses that as a place to begin. It then traverses the graph until it finds a word that is likely to be the end of a sentence.

In order to do this, some assumptions are made, and input data undergoes heavy processing prior to being inserted into the graph. Multiple regular expressions are used to parse undesirable or unusable characters out of the input, as well as garbage words. Sentences are assumed to end with a period, exclamation mark, or question mark; otherwise a sentence will be treated as a fragment. Sentences are generally limited to a certain length (cycles cannot be prevented due to the nature of the English language, so this is the only practical safeguard against them causing a sentence of infinite length).

3. Implementation

The Blather project is divided into three packages:

- blather.core - Contains the bulk of the engine. It is composed of the following classes:
 - Word - This class represents a single vertex on the graph, containing a word and its links to other words, as well as data about the word, such as how often it is at the beginning and end of a sentence, how often it is capitalized, and how strong the links to other words are. In essence, it has instance data and a list of WordLinks (which contain a reference to another word, and a link strength).
 - Sentence - This consists of a singly-linked list of Words, used both during the learning process (the Reader creates Sentences for the

Learner, both discussed below), and in the actual generation of sentences (allows adding words one by one).

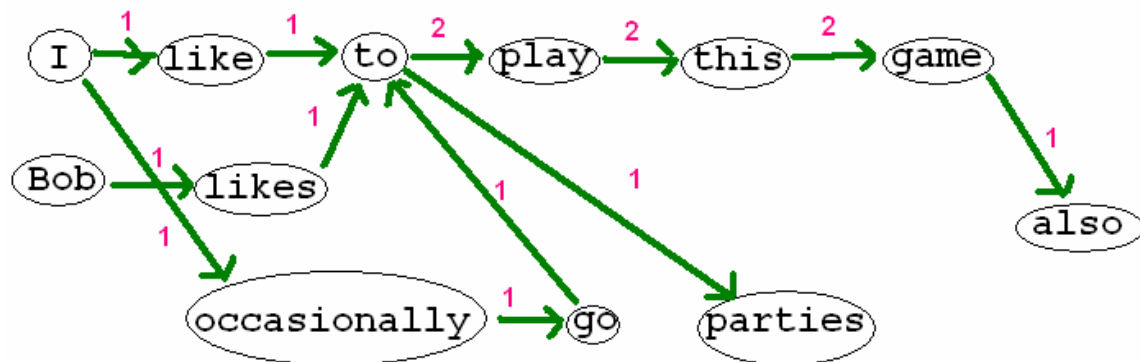
- Learner - This object, created by a Brain, allows one to read data into the Brain by feeding it strings or filenames. It will read in the data, parse it, and add/modify Word nodes in the graph as needed.
- Speaker - This class, also created by a Brain, generates Sentence objects based on a set of “seed” words, which act as a starting point in the graph from the sentences generated.
- Brain - This class represents both the entry point to the blather engine and the actual adjacency list graph of words itself. It contains a public Learner and Speaker object that can be accessed to modify the brain. It can also be saved to and loaded from files using Memory. Brains cannot have words removed once they are learned; Brains must be re-created when you want to start over.
- blather.io - This package contains classes related to reading and saving files. It includes:
 - Reader - This class acts much like java.util.Scanner. It takes in a string or a filename of a file containing sentences. It parses it and converts it to Sentence objects that the Learner can then feed into the Brain word by word.
 - Memory - This function uses the java serialization API to save Brain objects to file in their entirety and reconstitute them upon request.
- blather.gui - This package contains the Blather console and GUI interfaces (BlatherConsole and BlatherGUI respectively).

4. Examples

After adding the following three sentences:

- *I like to play this game.*
- *Bob likes to play this game also.*
- *I occasionally go to parties.*

...the Brain graph would look like this:



Links are shown in green, and the links' weight is marked in pink. As one can see from this graph, several other strings of words are possible, though some are less likely than others. "Bob likes to parties" is possible (not to mention grammatically incorrect), but this choice is less likely than the other option after "to", which is "play". The idea is that after learning enough, the choices the Speaker makes in selecting words will become more logical and make more sense as the probabilities of certain choices increase relative to that of other possible choices.

Possible pitfalls of this are that without learning material without several overlapping words, the sentences may be fairly repetitive. But with enough source material the results should be fairly interesting.

Here is an example conversation:

```
Welcome to Blather!
[Mode changed to Learner Mode]
You: Hello computer! my name is Katie.
Blather: My name is Katie.
You: That's a shocking coincidence!
Blather: That's a shocking coincidence!
You: Wow, you're a really funny guy.
Blather: really funny guy.
You: Hey! Stop repeating me!
Blather: Stop repeating me!
You: I'm not repeating you! You are repeating me!
```

Blather: I'm not repeating me!
You: HA! I broke you!
Blather: I broke you.
You: No you didn't!
Blather: No You.
You: No you didn't!
Blather: No you.
You: Silly Blather!
Blather: Silly Blather!
You: You just called yourself silly.
Blather: you are repeating me!
You: No I am not! You are silly!
Blather: No you just called yourself Silly!
You: Oh man, you crack me up.
Blather: Oh man, You.

5. The interface

The blather interfaces can be run on the console after all classes are compiled using: “java blather.gui.BlatherGUI” in the bin directory (just outside the ‘blather’ subfolder). Make sure that bg2.gif is in the working directory or it will fail to start. The console interface can be run using “java blather.gui.BlatherConsole”.