Recurrence Quantification as a Framework for Statistical Natural Language Processing

Rick Dale (rdale@ucmerced.edu)
Cognitive & Information Sciences
University of California, Merced

1. Introduction

All natural systems change in time, and their patterns of change can provide insight into how systems are organized and how they work. In the linguistic case, for example, studying the order in which words occur provides fodder for whole paradigms for studying syntax. From generative grammar (REF) to statistical approaches (REF), all proffer insight into the underlying nature of the cognitive system. In this paper, a time-based method of analysis is described and adapted for the analysis of text, called *recurrence quantification analysis* (RQA). It provides statistical descriptions of how a text is ordered. In this brief introduction, I offer some motivating verbiage for RQA, then follow with some specific derivations and demonstrations, and discuss how it might be a useful framework for conducting certain kinds of NLP.

RQA is a method commonly used to study noisy and continuously varying physical systems (REF). It was here that RQA was first introduced and developed (REF). RQA provides a suite of measures to describe the dynamic patterns present in a time series. Some have referred to it as a form of "generalized cross-correlation" (REF). It is used in two general and closely related ways. The first is to estimate invariant characteristics of a dynamic system. The second is to obtain more general statistical measures to describe systems or compare systems to each other.

In the former case, RQA is used as a tool that can precisely determine the type of system that is being studied. For example, measures from RQA can be used to estimate the control parameters of a dynamical system that is being simulated. Researchers may draw a time series from, say, a logistic map or Lorenz attractor using particular parameters on their defining equations. Researchers can then explore how RQA can be used to recover those parameters from a single short time series (e.g., REF). In this way, RQA contains precise information about a system from which a time series originates. Researchers have studied the conditions under which such information can be recovered (REF).

In the latter case, RQA is used for similar purposes, but the researcher remains more agnostic about the specific underlying properties of the system. Instead, this use of RQA seeks more general characteristics. Is one system more or less regularly structured in time? How prolonged are a system's persistent states or trajectories? What is the most stable regime the system exhibits in time? These questions have been prominently posed in psychology, for example, in characterizing postural dynamics under different conditions. Posture is a noisy and continuously varying signal, and RQA has allowed researchers to explore postural stability in various ways (e.g., REF).

RQA can be adapted to analyze patterns in discretely changing symbolic systems, such as a text (REF). Text has a proxy for time in the ordering of characters and words and so on (REF). A simple adaptation of standard RQA permits similar types of analysis of text as in the physical systems. We can quantify the dynamic structure present in sequentially ordered units, how regularly structured they are, and so on. In addition, we can utilize dynamical systems concepts in ways that may provide a useful theoretical terminology for studying texts. RQA provides a way of thinking of text in terms of time that may be a useful supplement to other kinds of NLP analysis.

In what follows I first briefly describe the way RQA is used in the most common case, on continuously varying physical systems. Following this, I show how it can be very easily adapted for the discrete case, and showcase it with text. I then show equivalence between RQA and some common NLP techniques, such as *n*-gram models. RQA can be seen as both a restricted subset of some NLP techniques, but in that respect, a generalized version of some models. I conclude with some discussion of resources available for working with recurrence (REF).

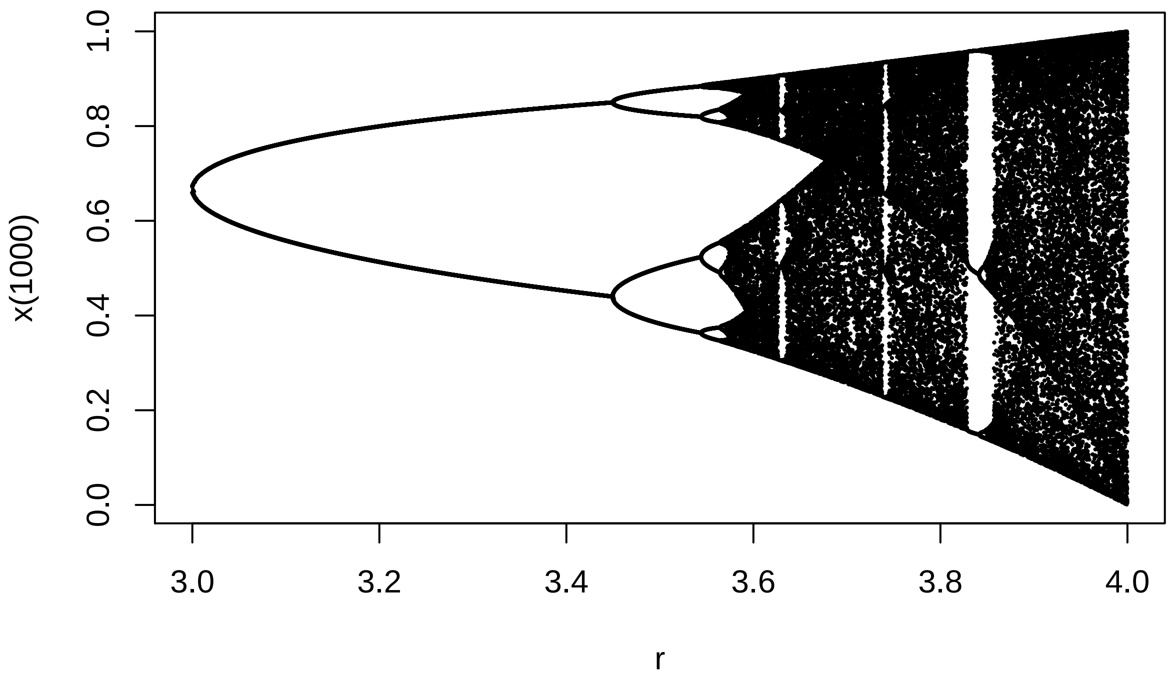
2. Brief Introduction to RQA

Some simplifying assumption are made here for the sake of presentation, but a very clear and freely available introduction to RQA can be found in REF. As noted above, RQA has been used to study dynamical systems. Researchers take a model system, and show that RQA measures can recover properties of that system. A prominent example of this is found in REF. They use the logistic map (REF). The logistic map is a discrete dynamical system defined by a one-dimensional state variable $x(t)$:

$$x(t+1) = rx(t)(1 - x(t))$$

The control parameter of this simple system is r, and it is assumed that $x(t) \in [0, 1]$. $x(t)$ has stable properties when t goes to ∞ when r is smaller than approximately 3. At that point, the system "bifurcates" and $x(t)$ stabilizes on two oscillating values. The system continues to bifurcate at specific junctures of the r parameter until $r = 3.56$ or so, at which point the system is said to exhibit an onset to chaos. It is outside the scope of the current paper, but this very simple system exhibits a wide range of interesting properties that can be used to demonstrate dynamical systems concepts. Fig. 1 shows how $x(t)$ behaves when $x(0) \in U(0, 1)$ and $t = 1,000$.[1]

```
plot(valuesR,xt,cex=.15,xlab='r',ylab='x(1000)')
```



REF shows that these patterns, such as the onset of chaos, can be signalled by analysis of simple shor time series using RQA. A convenient way to understand RQA is an analysis based on two stages. The first stage is producing a matrix referred to as a recurrence plot (*RP*). The second is to quantify the points on that plot, thus rendering RQA. The *RP* is constructed looking finding *recurrences* in the system.

[1]For readers interested in this system and its underlying properties, the best known such analysis is found in REF.

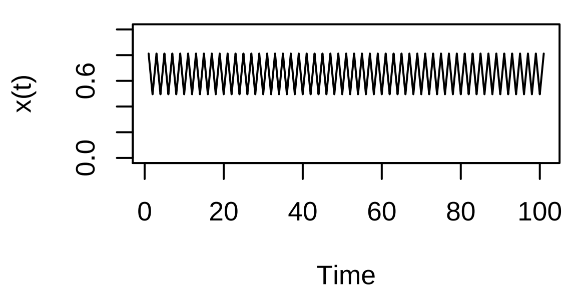
The RP is the set of points (i, j) such that $x(i)$ and $x(j)$ are within a certain distance ϵ of each other: $RP = \{(i, j)|D(x(i), x(j)) < \epsilon\}$. D is a metric over the system's states, and this general equation defines how to construct an RP for any time series. There are specific practices for determining the threshold ϵ, and other distances in this analysis, outside the scope of the present paper (see REF for a very thorough recent description). When considering $x(t)$ and extracting, say, a time series of 250 elements from $t = 900$ to $t = 1,000$, we have the following example RP if we take ϵ to be 5% of the range of the variable (an arbitrary convenience for presentation: $\epsilon = .05$). Consider two time series, one where $r = 3.25$ and another where $r = 3.6$. As you can see below, the RP for each of these time series is radically different, as would be anticipated from the influence of the control parameter at these values – one is bifurcating regularly between two values (3.25), and the other has the system in a chaotic regime (3.6).

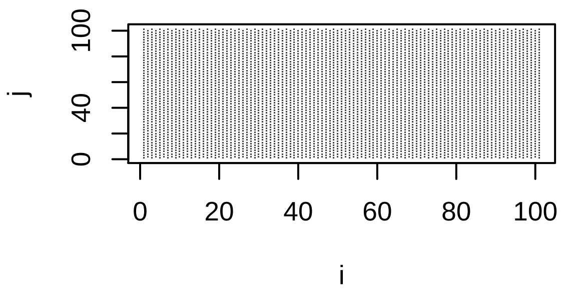
```
library(crqa)
```

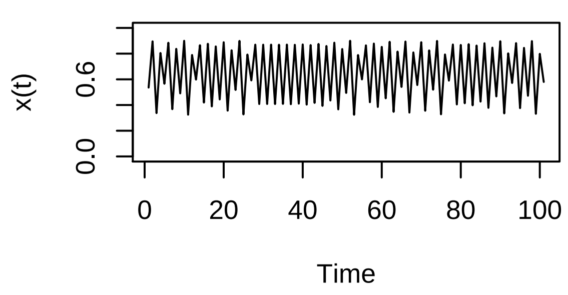
```
## Loading required package: Matrix
## Loading required package: tseriesChaos
## Loading required package: deSolve
## Loading required package: fields
## Loading required package: spam
## Loading required package: grid
## Spam version 1.0-1 (2014-09-09) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
##
## Attaching package: 'spam'
##
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
##
## Loading required package: maps
## Loading required package: pracma
##
## Attaching package: 'pracma'
##
## The following object is masked from 'package:deSolve':
##
##     rk4
##
## The following objects are masked from 'package:Matrix':
##
##     expm, lu, tril, triu
```

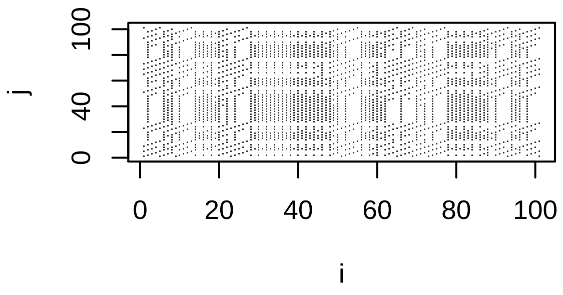
```
valuesR = 3.25
xt = runif(1)
for (i in 1:1000) {
  xt = c(xt,valuesR*xt[length(xt)]*(1-xt[length(xt)]))
}
par(mfrow=c(2,2))
plot(xt[900:1000],xlab='Time',ylab='x(t)',cex=.25,ylim=c(0,1),type='l')

rqaResults = crqa(xt[900:1000],xt[900:1000],1,1,0,.05,0,2,2,0)
```

```
RP = as.matrix(rqaResults$RP)
ij = which(RP==1,arr.ind=T) # get coordinates
plot(ij[,1],ij[,2],
     xlab='i',ylab='j',
     pch=15,cex=.1,col=rgb(.25,.25,.25))

valuesR = 3.6
xt = runif(1)
for (i in 1:1000) {
  xt = c(xt,valuesR*xt[length(xt)]*(1-xt[length(xt)]))
}
plot(xt[900:1000],xlab='Time',ylab='x(t)',cex=.25,ylim=c(0,1),type='l')

rqaResults = crqa(xt[900:1000],xt[900:1000],1,1,0,.05,0,2,2,0)
RP = as.matrix(rqaResults$RP)
ij = which(RP==1,arr.ind=T) # get coordinates
plot(ij[,1],ij[,2],
     xlab='i',ylab='j',
     pch=15,cex=.1,col=rgb(.25,.25,.25))
```


x(t)
Time


j
0
40
100
0
20
40
60
80
100
i


x(t)
Time


j
0
40
100
0
20
40
60
80
100
i