Extracting Patterns and Relations

from the World Wide Web

Sergey Brin

Computer Science Department

Stanford University

sergey@cs.stanford.edu

Abstract. The World Wide Web is a vast resource for information. At the same time it is extremely distributed. A particular type of data such as restaurant lists may be scattered across thousands of independent information sources in many di erent formats. In this paper, we consider the problem of extracting a relation for such a data type from all of these sources automatically. We present a technique which exploits the duality between sets of patterns and relations to grow the target relation starting from a small sample. To test our technique we use it to extract a relation of (author,title) pairs from the World Wide Web.

* Introduction

The World Wide Web provides a vast source of information of almost all types, ranging from DNA databases to resumes to lists of favorite restaurants. However, this information is often scattered among many web servers and hosts, using many di erent formats. If these chunks of information could be extracted from the World Wide Web and integrated into a structured form, they would form an unprecedented source of information. It would include the largest international directory of people, the largest and most diverse databases of products, the greatest bibliography of academic works, and many other useful resources.

There has been considerable work on integrating a number of information sources using specially coded wrappers or lters [Tsi,MOS97]. However, these can be time-consuming to create and are usually used for tens, not thousands of sources. In this paper, we address the problem of extracting a relation from the thousands of sources that may hold pieces of the relation on the World Wide Web. Our goal is to discover information sources and to extract the relevant in-formation from them either entirely automatically, or with very minimal human intervention.

In this paper, we consider the problem of extracting a relation of books { (author,title) pairs from the Web. Intuitively, our solution works as follows. We begin with a small seed set of (author, title) pairs (in tests we used a set of just ve books). Then we nd all occurrences of those books on the Web. From these occurrences we recognise patterns for the citations of books. Then we search the Web for these patterns and nd new books. We can then take these books and

nd all their occurrences and from those generate more patterns. We can use these new patterns to nd more books, and so forth. Eventually, we will obtain a large list of books and patterns for nding them.

* The Duality of Patterns and Relations

The method we propose is called DIPRE - Dual Iterative Pattern Relation Ex-pansion. It relies on a duality between patterns and relations which we explain below.

2.1 The Problem

Here we de ne our problem more formally:

Let D be a large database of unstructured information such as the World Wide Web. Let R = r1; :::; rn be the target relation. Every tuple, t, of R occurs in one or more times in D. Every such occurrence consists of all the elds of t, represented as strings, occurring in close proximity to each other in D (in the case of the Web, this means all the elds are near each other, on the same Web page).

In the test problem we examine in this paper, the target relation R is the set of books { (author, title) pairs that occur on the Web. Clearly, this is not well de ned. However, given a potential author and title and where they are mentioned on the Web, a human can generally tell whether this is a legitimate

book. 0

If we compute an approximation, R0 of R then the coverage is jRj \jRj and the

R

error rate is jR0 ,Rj . Our goal is to maximize coverage and minimize the error

jR0 j

rate. However, a low error rate is much more critical than high coverage. Given a su ciently large database, D, a recall of just 20% may be acceptable. However, an error rate over 10% would likely be useless for many applications.

Typically, we cannot actually compute R. Therefore, we cannot not know the precise values of coverage and error rate. However, we can sample the error rate by having a user check random elements of R0. Coverage is much more di cult to estimate.

2.2 Patterns

Intuitively, a pattern matches one particular format of occurrences of tuples of the target relation. Ideally the pattern is speci c enough not to match any tuples that should not be in the relation, however, in practice a few false positives may occur. Patterns may have various representations. In our work we used a very limited class of regular expressions. More formally:

Let p be a pattern. Then MD(p) is the set of tuples that match p in D and jpjD is the number of elements in MD(p). Then the coverage of p,CD (p; R) = jMD(p) \ Rj=jRj and the error rate of p is ED(p; R) = jMD(p) , Rj=jMD(p)j.

For a set of patterns, P = p1; :::; pk, we de ne MD (P) = Sp2P MD(p). We extend CD (P; R) and ED (P; R) analogously. Alternative de nitions of MD (P ) may require a tuple to match multiple patterns (see Section 6).

2.3 Pattern Relation Duality

An important observation is that given a set of patterns, P with high coverage and low error rate, we can construct a very good approximation to R simply by nding all matches to all the patterns. Thus, given a good set of patterns, we can build a good set of tuples. However, we also wish to have the converse property - given a good set of tuples, we can build a good set of patterns. We can do this by nding all occurrences of the tuples in D and discovering similarities in the occurrences. The combination of the ability to nd tuples from patterns and patterns from tuples gives us great power and is the basis for the technique we propose in this paper.

* Dual Iterative Pattern Relation Extraction

Dual Iterative Pattern Relation Extraction - DIPRE is a technique for extracting relations which makes use of pattern-relation duality. It works as follows:

1. R0 Sample

Start with a small sample, R0 of the target relation. This sample is given by the user and can be very small. In our tests, we used a list of ve books with authors.

1. O FindOccurrences(R0; D)

Then, nd all occurrences of tuples of R0 in D. In our experiments, these were nearby occurrences of the author and the title of a book in text. Along with the tuple found, keep the context of every occurrence (url and surrounding text).

1. P GenPatterns(O)

Generate patterns based on the set of occurrences. This is the tricky part of the algorithm. Roughly speaking, this routine must generate patterns for sets of occurrences with similar context. The patterns need to have a low

error rate, so it is important that they are not overly general. The higher the coverage of the patterns the better. However, a low coverage can be compensated for with a larger database.

4. R0 MD(P ). Search the database for tuples matching any of the patterns.

5. If R0 is large enough, return. Else go to step 2.

3.1 Controlling Expansion

The above process is not necessarily very stable and may stray away from R. In particular, several bogus tuples in MD (P) can lead to several bogus patterns in

* in the next iteration. This in turn can cause a whole slew of bogus tuples. For

this reason the GenPatterns routine must be careful to minimize the amount of damage caused by a potential bogus tuple (or several small tuples). Another measure of safety is to de ne MD(P ) more stringently so as to require tuples to match multiple patterns in P . This second measure has not been necessary in the tests we have performed but may be necessary in future tests. Finally, various threshholds may need to uctuate as the relation expands.

* Finding Authors and Titles

For our experiments we chose to compute a relation of (Author,Title) pairs from the World Wide Web. This problem lends itself particularly well to DIPRE because there are a number of well-known books which are listed on many web sites. Many of the web sites conform to a reasonably uniform format across the site.

4.1 Patterns for Books

In order to use DIPRE to nd books, it is necessary to de ne what patterns consist of. The de nition of a pattern largely determines the success of DIPRE. However, for our tests we used a very simple de nition of a pattern. It requires further investigation to determine whether more sophisticated de nitions of pat-terns work better.

We de ned a pattern as a ve-tuple: (order, urlpre x, pre x, middle, su x) where order is a boolean value and the other attributes are strings. If order is true, an (author,title) pair matches the pattern if there is a document in the collection (the WWW) with a URL which matches urlprefix\* and which con-tains text that matches the regular expression:

\*prefix, author, middle, title, suffix\*

The author is restricted to:

[A-Z][A-Za-z .,&]5;30[A-Za-z.]

The title is restricted to:

[A-Z0-9][A-Za-z0-9 .,:'#!?;&]4;45[A-Za-z0-9?!]

If order is false, then the title and author are switched.

4.2 Occurrences

We also have to de ne how an occurrence is structured since it should have a correspondance to the de nition of a pattern. An occurrence of an (author,title) pair consists of a seven-tuple:

(author, title, order, url, pre x, middle, su x)

The order corresponds to the order the title and the author occurred in the text. The url is the URL of the document they occurred on. The pre x consists of the m characters (in tests m was 10) preceding the author (or title if the title was

rst). The middle is the text between the author and title and the su x consists of the m characters following the title (or author).1

4.3 Generating Patterns for Books

An important component of the DIPRE procedure is the GenPatterns routine which takes a set of occurrences of books and converts them into a list of patterns. This is a nontrivial problem and there is the entire eld of pattern recognition devoted to solving the general version of this problem. For our purposes, however, we use a simple set of heuristics for generating patterns from occurrences. As long as there are few false positives (patterns that generate nonbooks) this is su cient. Each pattern need only have very small coverage since the web is vast and there are many sources of information so the total coverage of all the patterns can still be substantial.

Suppose we are given a set of occurrences and we wish to construct a speci c a pattern as possible that matches all of them. We can do this as follows:

1. Verify that the order and middle of all the occurrences is the same. If not, it is not possible to generate a pattern to match them all. Set outpattern.order and outpattern.middle to order and middle respectively.
2. Find the longest matching pre x of all the urls. Set outpattern.urlpre x to that pre x.
3. Set outpattern.pre x to the longest matching su x of the pre x's of the occurrences.
4. Set outpattern.su x to the longest matching pre x of the su x's of the occurrences.

We denote this routine GenOnePattern(O).

Pattern Speci city A pattern generated like the above can be too general or too speci c. We are not concerned about it being too speci c since there will be many patterns generated and combined there will be many books. However, the pattern may be too general and may produce many nonbooks.

To combat this problem we attempt to measure the speci city of the pattern. The speci city of a pattern p roughly corresponds to ,log(P (X 2 MD (p))) where X is some random variable distributed uniformly over the domain of tuples of R.2 For quick computation, we used the following formula for the speci city of a pattern (jsj denotes the length of s):

speci city(p) = jp:middlejjp:urlpre xjjp:pre xjjp:su xj

* The pre x and su x could actually be less than m characters if the line ends or starts close to the occurrence but this is a restriction of the current implementation

and it is unclear whether it has a positive or negative e ect.

* If the domain is in nite like the space of all strings, the uniform distribution may not be sensible and a di erent distribution should be used.

We reject any patterns with too low a speci city so that overly general pat-terns aren't generated. More speci cally, we insist that speci city(p)n > t where n is the number of books with occurrences supporting the pattern p and t is a threshhold. This ensures that all the strings of a pattern are nonempty (other-wise the speci city is zero). Also we require that n > 1 since basing a pattern on one example is very error-prone.

Algorithm for Generating Patterns Here, we present the algorithm for Gen-Patterns(O). It takes advantage of the algorithm GenOnePattern(O) introduced in Section 4.3.

1. Group all occurrences o in O by order and middle. Let the resulting groups be O1; :::Ok.
2. For each group Oi, p GenOnePattern(Oi). If p meets the speci city re-quirements then output p. Otherwise:

{ If all o in Oi have the same URL then reject Oi.

{ Else, separate the occurrences o in Oi into subgroups grouped by the character in their urls which is one past p.urlpre x. Repeat the procedure in step 2 for these subgroups.

This routine uses a simple further subdivision based on the url when the pattern generated is not su ciently speci c. One can also imagine using the pre x or the su x.

We have described a simple technique for generating patterns from lists of occurrences books. One can imagine far more sophisticated techniques and this is the subject of further research. However, as is indicated by the results (Section 5) even this simple scheme works well.

4.4 Performance Issues

There are two very demanding tasks DIPRE - nding occurrences of books given a long list of books and nding pattern matches given a list of patterns. Both of these operation must take place over a very large database of Web documents.

For the rst task, nding occurrences of books, we rst pass the data through two fgrep lters. One only passes through lines that contained a valid author and the other only passes through lines that contained a valid title. After this it is the task of a program written in Python to actually check that there are matching authors and titles in the line, identify them and produce occurrences as output. Several alternative approaches involving large regular expressions in Flex and in Python were attempted for this purpose but they quickly exceeded various internal bounds.

For the second task, we use just a Python program. Every pattern is trans-lated into a pair of regular expressions, one for the URL, and one for the actual occurrence. Every URL is rst tested to see which patterns apply to it. Then the program tests every line for the relevant regular expressions. This approach is quite slow and needs to be improved. Future versions are likely to use Flex or

the rex C library. This task can be made somewhat easier by targeting just the URL's which match the patterns and we made some attempt to do this. How-ever, the data is not structured to make that completely trivial and we wish the techniques we develop to be general enough to be able to handle no restrictions on URL's.

The generation of patterns from occurrences is not much of a performance issue at this point in time because there are only thousands of occurrences gen-erated. As larger tests are run this will become more important. Currently, the occurrences are sorted using gsort by order and middle. Then a Python pro-gram reads through the resulting list and generates patterns as described in Section 4.3.

* Experiments

While the experiments performed so far have been very limited, due to time constraints they have produced very positive results. Many more experiments are in progress.

5.1 Web Data Used in Experiments

For data we used a repository of 24 million web pages totalling 147 gigabytes. This data is part of the Stanford WebBase and is used for the Google search engine [BP] and other research projects. As a part of the search engine, we have built an inverted index of the entire repository.

The repository spans many disks and several machines. It takes a considerable amount of time to make just one pass over the data even without doing any substantial processing. Therefore, in these we only made passes over subsets of the repository on any given iteration.

An important note for this project is that the repository contains almost no web pages from Amazon [Ama]. This is because their automatically generated urls make crawling di cult.

5.2 Pattern Relation Expansion

|  |  |
| --- | --- |
| Isaac Asimov | The Robots of Dawn |
| David Brin3 | Startide Rising |
| James Gleick | Chaos: Making a New Science |
| Charles Dickens | Great Expectations |
| William Shakespeare | The Comedy of Errors |

Fig. 1. Initial sample of books.

|  |  |
| --- | --- |
| URL Pattern | Text Pattern |
| www.sff.net/locus/c.\* | <LI><B>title</B> by author ( |

~

dns.city-net.com/lmann/awards/hugos/1984.html <i>title</i> by

~

dolphin.upenn.edu/dcummins/texts/sf-award.htm author || title

author (

|| (

Fig. 2. Patterns found in rst iteration.

We started the experiment with just 5 books (see Figure 1). These produced 199 occurrences and generated 3 patterns (see Figure 2). Interestingly, only the rst two of the ve books produced the patterns because they were both sci-ence ction books. A run of these patterns over matching URL's produced 4047 unique (author,title) pairs. They were mostly science ction but there were some exceptions. (See Figure 3.

|  |  |
| --- | --- |
| H. D. Everett | The Death-Mask and Other Ghosts |
| H. G. Wells | First Men in the Moon |
| H. G. Wells | Science Fiction: Volume 2 |
| H. G. Wells | The First Men in the Moon |
| H. G. Wells | The Invisible Man |
| H. G. Wells | The Island of Dr. Moreau |
| H. G. Wells | The Science Fiction Volume 1 |
| H. G. Wells | The Shape of Things to Come: The Ultimate Revolution |
| H. G. Wells | The Time Machine |
| H. G. Wells | The War of the Worlds |
| H. G. Wells | When the Sleeper Wakes |
| H. M. Hoover | Journey Through the Empty |
| H. P. Lovecraft & August Derleth | The Lurker at the Threshold |
| H. P. Lovecraft | At the Mountains of Madness and Other Tales of Terror |
| H. P. Lovecraft | The Case of Charles Dexter Ward |
| H. P. Lovecraft | The Doom That Came to Sarnath and Other Stories |

Fig. 3. Sample of books found in rst iteration.

A search through roughly 5 million web pages found 3972 occurrences of these books. This number was something of a disappointment since it was not a large blowup as had happened in the rst iteration. However, it would have taken at least a couple of days to run over the entire repository so we did not attempt to generate more. These occurrences produced 105 patterns, 24 of which had url pre xes which were not complete urls. A pass over a couple million urls produced 9369 unique (author, title) pairs. Unfortunately, there were some bogus books among these. In particular, 242 of them were legitimate titles but had an author of \Conclusion". We removed these from the list. This was the only manual intervention through the whole process. In future experiments, it would

be interesting to see whether leaving these in would produce an extraordinary amount of junk.

For the nal iteration, we chose to use the subset of the repository which con-tained the work books. This consisted of roughly 156,000 documents. Scanning for the 9127 remaining books produced 9938 occurrences. These in turn gen-erated 346 patterns. Scanning over the same set of documents produced 15257 unique books with very little bogus data. (See Figure 4)

This experiment is ongoing and hopefully, a larger list of books will be gen-erated soon. The current one is available online [Bri].

5.3 Quality of Results

To analyse the quality of the results, we picked twenty random books out of the list and attempted to verify that they were actual books by searching on Amazon [Ama], the Visa Shopping Guide for books [Vis], the Stanford online library catalog, and the Web.4 As a measure of the quality of the results, 19 of the 20 were all bona de books. The remaining book was actually an article - \Why I Voted for a User Car", by Andrew Tobias.

The big surprise was that a number of the books were not found in some or all of the sources except for the Web. Some of these books were online books; some were obscure or out of print; some simply were not listed on some sites for no apparent reason. In total, 5 of the 20 books were not on Amazon which claims to have a catalog of 2.5 million books.

Other than the article mentioned above, there are a few visible problems with the data. Some books are mentioned several times due to small di erences such as capitalization, spacing, how the author was listed (for example \E.R. Burroughs" versus \Edgar Rice Burroughs"). Fortunately, however, authors are quite particular about how their name is listed and these duplications are limited. In several cases, some information was appended to the author's name such as publication date.

* Conclusions

Our general goal is to be able to extract structured data from the entire World Wide Web by leveraging on its vastness. DIPRE has proven to be a remarkable tool in the simple example of nding lists of books. It started with a sample set of 5 books and expanded it to a relatively high quality list of over 15,000 books with very minimal human intervention. The same tool may be applied to a number of other domains such as movies, music, restaurants, and so forth. A more sophisticated version of this tool is likely to be able to extract people directories, product catalogs, and more.

* Unfortunately, the Library of Congress search system was down at the time of these tests.

|  |  |
| --- | --- |
| Henry James | The Europeans |
| Henry James | The Golden Bowl |
| Henry James | The Portrait of a Lady |
| Henry James | The Turn of the Screw |
| Henry James | Turn of the Screw |
| Henry John Coke | Tracks of a Rolling Stone |
| Henry K. Rowe | Landmarks in Christian History |
| Henry Kisor | Zephyr |
| Henry Lawson | In the Days When the World Was Wide |
| Henry Longfellow | The Song of Hiawatha |
| Henry Miller | Tropic of Cancer |
| Henry Petroski | Invention On Design |
| Henry Petroski | The Evolution of Useful Things |
| Henry Roth | Call It Sleep |
| Henry Sumner Maine | Ancient Law |
| Henry Tuckerman, Lindsay, Phila | Characteristics of Literature |
| Henry Van Dyke | The Blue Flower |
| Henry Van Dyke, Scrib | Days O |
| Henry Van Loon | Life and Times of Pieter Stuyvesant |
| Henry Wadsworth Longfellow | Paul Revere's Ride |
| Henry Wadsworth Longfellow | Evangeline |
| Henry Wadsworth Longfellow | The Song of Hiawatha |
| Herbert Donald | Lincoln |
| Herbert M. Hart | Old Forts of the Northwest |
| Herbert M. Mason, Jr | The Lafayette Escadrille |
| Herbert R. Lottman | Jules Verne: An Exploratory Biography |
| Herbert Spencer | The Man Versus the State |
| Herman Daly | For the Common Good |
| Herman Daly | Valuing the Earth |
| Herman E. Kittredge | Ingersoll: A Biographical Appreciation |
| Herman Haken | Principles of Brain Functioning |
| Herman Hesse | Demian |
| Herman Hesse | Siddhartha |
| Herman Hesse | Sidharta |
| Herman Melville | Bartleby, the Scrivener |
| Herman Melville | Billy Budd |
| Herman Melville | Billy Budd |
| Herman Melville | Moby Dick |
| Herman Melville | The Con dence Man |
| Herman Melville | The Encantadas, or Enchanted Isles |
| Herman Melville | Typee: A Peep at Polynesian Life |
| Herman Weiss | Sunset Detectives |
| Herman Wouk | War And Remembrance |
| Hermann Hesse | Klingsor's Last Summer |
| Hermann Hesse | Knulp |
| Hermann Hesse | Rosshalde |
| Hermann Hesse | Strange News From Another Star |
| Herodotus | Histories |
| Herodotus | The Histories |
| Herodotus | The History of Herodotus |
| Herschel Hobbs | Pastor's Manual |
| Hetschel | First Stage: Moon |
| Hiaasen | Stormy Weather |
| Hilaire | Surivals and New Arrivals |
| Hilaire | The Great Heresies |
| Hilary Bailey | Cassandra: Princess of Troy |
| Hilary Norman | The Key to Susanna |
| Hilbert Schenck | Chronosequence |
| Hilbert Schenck | The Battle of the Abaco Reefs |
| Hilda Conkling | Poems by a Little Girl |
| Hilda Hughes | Shudders |
| Hilda Hughes | When Churchyards Yawn |
| Hillerman | A Thief of Time |
| Hillerman | Skinwalkers |
| Hillerman | Talking God |
| Hiram Corson | Introduction to Browning |
| Hjalmar Hjorth Boyesen | Boyhood in Norway |
| Hjalmar Hjorth Boysen | Tales From Two Hemispheres |
|  |  |

Fig. 4. Sample of books in the nal list.

6.1 Scalability and Steady State

There are several challenges to the scalability of this method. One is the per-formance required to scan for large numbers of patterns and tuples over a huge repository. Improvements in the underlying algorithms and implementation are likely to solve this problem in the very near future.

A potentially more di cult obstacle is whether DIPRE can be kept from diverging from the target as it expands the relation. For example, since it really used only the two science ction books which were in the seed sample, why did it not produce a large list of science ction books. Clearly, it gravitated to a compilation of all books and even a few scatterred articles managed to enter the relation. Keeping this e ect under control as the relation expands is nontrivial but there are several possibilities.

Connection to Singular Value Decomposition One possibility is to rede-ne of MD(P ) to require multiple patterns to match a tuple. A more extreme version of this is to assign a weight to every tuple and pattern. A matching tuple is assigned a weight based on the weights of the patterns it matches. A generated pattern is assigned a weight based on the weights of the tuples which match it. If this is done linearly, this technique breaks down to a singular value decomposi-tion of the tuple-pattern matrix (multiplied by its transpose). This is analogous to Latent Semantic Indexing [DDF+90] which is done on the document-word matrix. In this case, the eventual steady state is the dominant eigenvector. Un-fortunately, this is independent of the initial sample which is clearly not desirable. Nonetheless, the relationship to LSI is compelling and bears further investiga-tion.

The independence of the steady state from the initial state above may also be a problem even without the use of weights. There are several possible solutions. One is to run only through a limited number of iterations like we demonstrated in this paper. Another solution is to make sure that the transformation of tuples to patterns to tuples is nonlinear and has some local steady states which depend on the initial state. This can be accomplished through the use of the initial

sample R0 in the computation of GenPatterns. In this case, the user may also 0

provide an R , a list of counterexamples.

6.2 Implications of Automatic Extraction

One of the most surprising results of this experiment was nding books which were not listed in major online sources such as the book \Disbanded" by Douglas Clark [Cla] which is published online or \The Young Gardeners' Kalendar" by Dollie Radford [Rad04] an obscure work published in 1904. If the book list can be expanded and if almost all books listed in online sources can be extracted, the resulting list may be more complete than any existing book database. The generated list would be the product of thousands of small online sources as

opposed to current book databases which are the products of a few large infor-mation sources. Such a change in information ow can have important social rami cations.

References

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [Ama] | Amazon home page. http://www.amazon.com. | | | | |  |
| [BP] | Sergey Brin and | | Larry | Page. | Google search engine. | http://google. |
|  | stanford.edu. |  |  |  |  |  |
| [Bri] | Sergey Brin. | List of | | books. | http://www-db.stanford.edu/~sergey/ | |
|  | booklist.html. |  |  |  |  |  |
| [Cla] | Douglas Clark. Disbanded. Benjamin Press, 69 Hillcrest Drive, Bath Ba2 | | | | | |
|  | 1HD, UK. http://www.bath.ac.uk/~exxdgdc/poetry/library/di1.html. | | | | | |
| [DDF+90] | Scott Deerwester, Susan Dumais, Goerge Furnas, Thomas Landauer, and | | | | | |
|  | Richard Harshman. Indexing by latent semantic analysis. Journal of the | | | | | |
|  | American Society for Information Science, 41(6):391{407, 1990. | | | | | |
| [MOS97] | Workshop on management of semistructured data. http://www.research. | | | | | |
|  | att.com/~suciu/workshop-papers.html, May 1997. | | | | |  |
| [Rad04] | Dollie Radford. | | The | Young | Gardeners' Kalendar. | Alexander Mor- |
|  | ing, Ltd., London, 1904. http://www.indiana.edu/~letrs/vwwp/radford/ | | | | | |
|  | kalendar.html. |  |  |  |  |  |
| [Tsi] | Tsimmis home | page. | | http://www-db.stanford.edu/tsimmis/tsimmis. | | |
|  | html. |  |  |  |  |  |
| [Vis] | Visa shopping guide for books. http://shopguide.yahoo.com/shopguide/ | | | | | |
|  | books.html. |  |  |  |  |  |