Publishing Persian Linked Data; Challenges and Lessons Learned

Behshid Behkamal[1], Mohsen Kahani[1], Samad Paydar[1], Mahboobeh Dadkhah[1], Elaheh Sekhavaty[2]
[1]Web Technology Lab., Dept. of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[2]Payam Noor University of Tehran, Tehran, Iran
behkamal@stu-mail.um.ac.ir, kahani@um.ac.ir, samad.paydar@stu-mail.um.ac.ir, mb.dadkhah@stu-mail.um.ac.ir, sekhavaty@tehran.pnu.ac.ir

***Abstract* - This paper discusses the challenges of publishing Persian linked data based on an experience of publishing some academic data from Ferdowsi University of Mashhad dataset. By analyzing the experimental results of the project and classifying the problems, some publisher-oriented solutions are proposed to improve the quality of datasets on the web.**

Keywords- Linked Data; RDF; Organizational Data; Persian Dataset

I. INTRODUCTION

Linked Data movement has been integral to RDF publishing on the Web, emphasizing four basic principles: (i) use URIs as names for things; (ii) use HTTP URIs so that those names can be looked up; (iii) provide useful information when a look-up on that URI is made; and (iv) include links using external URIs [2].

Over the past few years, many web publishers have turned to RDF as a means of disseminating information in an open and machine-interpretable way, resulting in a "Web of Data" which now includes interlinked content exported from corporate bodies, biomedical datasets, governmental entities and organizational data.

Data of universities and their activities is important to many web users like students, researchers and teachers. Such data, if published as Linked Data and linked to appropriate datasets (e.g. general datasets like DBpedia, or special datasets like DBLP or ACM), can provide valuable benefits by enabling different scenarios of fullfiling users' information need. For instance, it can help students to search for professors or departments to apply, based on the professor's attributes or the properties of the department.

We herein discuss some problems and challenges of linked data, along with possible publisher-oriented approaches to improve the quality of structured, machine-readable and open data on the Web based on our experiences with "FUM-LD" project [9]. FUM-LD is a framework developed for publishing the data of Ferdowsi University of Mashhad (FUM) as Linked Data.

The paper structure is as follows: some related works are presented in section II. Different parts of the FUM-LD framework with analysis of the experimental results are discussed briefly in sections III. By performing this project, the problems of publishing Persian linked data are identified which are discussed in section IV. Finally, our future works are presented in section V.

II. RELATED WORK

Generally speaking, one of the main challenges in Linked data is interlinking between datasets. The links can either be set manually or generated by automated linking algorithms for large datasets. For the latter case, [5] have shown that simple interlinking algorithms produce rather poor results. Naive approaches trying to perform a simple literal lookup are likely to fail. When trying to interlink data from, for instance, the geographical domain with Geonames, it is possible to do a simple literal lookup using the search facility provided by Geonames. However, when querying for the city Vienna almost 20 results will be returned as there exist that many cities named Vienna around the world. Advanced approaches described in [5] are needed to disambiguate similar matches and finally create appropriate interlinks. Their algorithm was implemented within the GNAT tool and has been evaluated for interlinking music-related data sets.

In [10] the authors extract social graphs from online networks to investigate overlapping network fragments to link instances of persons from different information sources. Three alternative methods for computing graph similarity are discussed, including a low-level reasoning approach to investigate the implicit semantic similarity. Identified matches in separated graphs are resolved and the resulting links are in turn provided as a social graph.

With the growing amount of published data, integration issues also started to receive attention recently [11]. These systems abstract from schema-level issues and focus on finding co-referent instances, assuming their type and structure to be the same. RDF-AI [11] concentrates on data-level issues when combining datasets using the same schema. The algorithm uses string and linguistic (WordNet) similarity to calculate distance between literal property values and then uses the iterative graph matching algorithm to calculate distance between individuals [12].

In [13], authors discussed common errors in RDF publishing, their consequences for applications, along with possible publisher-oriented approaches to improve the quality of structured, machine-readable and open data on the Web. They provided discussions for some issues like issues relating to how data is found and accessed, parsing and syntax issues, reasoning issues, inconsistent data, and ontology hijacking, both from the perspectives of publishers and data consumers.

III. FUM-LD PROJECT

In this section, the process of publishing FUM-LD is briefly described in 4 steps as follows.

A. *Selecting Target Data*

Different educational and organizational web-based systems are being used at Ferdowsi university of Mashhad. Currently, these systems store their data in relational databases and publish parts of this data on the Web, using traditional approaches. After studying the FUM database, five important entities are selected consisting of faculties, departments, professors, papers (published by professors) and courses. TABLE I. shows the numbers of entities in FUM database which are selected to be published as linked data.

TABLE I. NUMBER OF ENTITIES

Entity	Count
Faculty	15
Department	89
Professor	845
Paper	9777
Course	5834
Total	16560

B. *Assigning URIs*

There are different approaches for assigning URIs to entities should be published. In FUM-LD, a simple schema is used for this purpose:

URI schema: http://wtlab.um.ac.ir/linkeddata/TYPE/ID

where TYPE is one of the strings 'faculties', 'departments', 'profs', 'papers' and 'courses' based on the type of the entity, and ID is the unique identifier of the entity in the database. For instance, http://wtlab.um.ac.ir/linkeddata/profs/kahani describes the resource corresponding to Mohsen Kahani.

C. *Publishing Data*

An overview of FUM-LD framework is shown in Figure 1. It is implemented in Java and consists of a repository and three core applications briefly introduced in the following subsections:

- RDFizer for generating RDF representation of the entities
- RDF2HTML for converting RDF representation of the entities to HTML
- vioDGenerator for creating vioD specification of FUM-LD

1) *RDFizer*

RDFizer extracts data from FUM relational database and creates an RDF file for describing each entity, and stores it in the repository. Different vocabularies are used in describing resources: FOAF[1] is used for describing personal information of professors and their social network (including other professors who are members of the same faculty and department). Dublin Core[2], BibTeX[3], and MarcOnt[4] are used for describing publications of professors. SKOS [6] subjects are used in describing courses, departments and faculties.

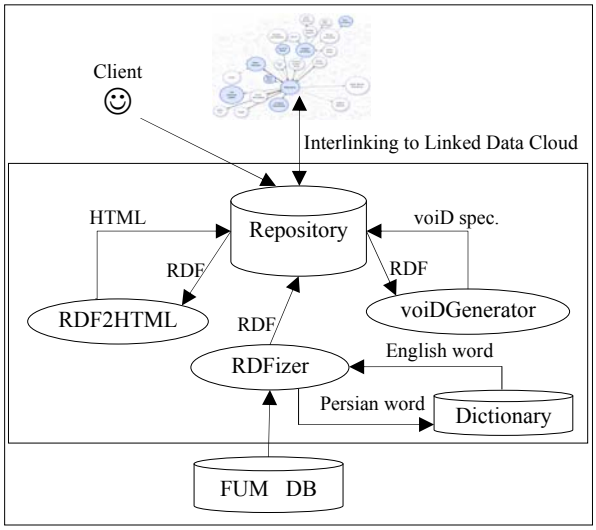

Figure 1. FUM-LD Framework

Linking FUM dataset to other external datasets consist of two steps. Since FUM is a Persian dataset, at first each term should be translated to English. For this purpose, a local dictionary is used to find appropriate equivalent of all terms. Then application automatically searches the external dataset for each English term using its SPARQL endpoint. This search is based on a number of empirical heuristics and simple SPARQL templates defined in RDFizer which are instantiated in runtime to perform the search.

2) *RDF2HTML*

In addition to RDF representation, FUM-LD framework generates human-friendly HTML representation of resources. RDF2HTML processes the RDF files in the repository and generates corresponding HTML files and stores them in the same repository.

3) *voiDGenerator*

The framework uses voiD [1] vocabulary to describe the published dataset. It is a vocabulary for describing RDF datasets in terms of their provenance, statistical, structural and licensing information. Using voiD to describe published datasets provides advantages from different points of view, such as trust, searching, ranking and selecting datasets [1], [7].

voiDGenerator processes RDF files in the repository and generates the voiD specification of the whole dataset as a single RDF file. In addition to some basic information about the dataset (e.g. its subject, definition, publication date, contributors, example resources …), this specification declares the main vocabularies used in describing the resources, number

[1] http://xmlns.com/foaf/spec/

[2] http://dublincore.org/

[3] http://www.bibtex.org/

[4] http://www.marcont.org/

of resources of type foaf:Person, total number of RDF triples, different subsets and linksets of the dataset.

D. Interlinking Data Resources

Currently, in most linked data publishing projects, interlinks between web datasets are generated entirely automatically, using heuristics to determine when two resources in two datasets identify the same object ([1], [5]). Providing links to other resources inside and outside the FUM-LD is an important issue in publishing this dataset and the RDFizer is responsible for generating such links.

1) Linking to Other Resources

Resources in FUM-LD are automatically linked to different LOD datasets. The faculty and department titles and course names are linked to related resources in DBpedia with owl:sameAs links. Countries, provinces and cities of the faculties and departments are linked to Geonames dataset by foaf:based_near predicate. Courses are linked to related terms in OpenCyc. Professors and their publications are linked to equivalent resources in DBLP and ACM. TABLE II. shows some statistics about these links.

TABLE II. SOME STATISTICS OF THE FUM-LD LINK SETS

Link set	Description	Count
1	Links to DBpedia Resources	4570
2	owl:sameAs links to DBpedia	1311
3	owl:sameAs links to DBLP	475
4	owl:sameAs links to ACM	38
5	skos:subject links to DBpedia	3708
6	skos:subject links to OpenCyc	449
7	Links to GeoNames resources	936

2) Interlinking to FUM-LD

In addition to links to external datasets, there are some internal links between different resources in the FUM-LD. For instance, each professor is linked to courses he/she teaches. As shown in Figure 2. there are five different subsets in FUM-LD. This figure shows existing links between these datasets. This interlinking helps user to browse the dataset easier.

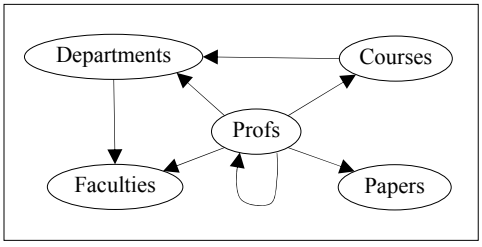

Figure 2. Interlinking of FUM-LD dataset

The total number of RDF triples in dataset is 317916, and there are 845 foaf:Person resources described. The FUM-LD consists of 5 subsets: Faculties, Department, Courses, Profs, and Papers. TABLE III. shows the number of links between these subsets.

TABLE III. LINKS BETWEEN DIFFERENT SUBSETS OF FUM-LD

Source subset	Target subset	Number of links	Link type example
Profs	Profs	15150	foaf:knows
Profs	Faculties	845	foaf:member
Profs	Departments	845	foaf:member
Profs	Courses	15447	dmcNS:teaches[5]
Profs	Papers	13110	foaf:maker, dc:creator
Courses	Departments	17502	dbpprop:reference[6]
Departments	Faculties	174	dc:ispartof

IV. CHALLENGES AND SOLUTIONS

Different problems and challenges are identified during FUM-LD project. Here, we discuss these problems and recommend some solutions to publishers. We begin with issues relating to linking and accessing other datasets; then we discuss data problems, Persian language challenges, and maintenance challenges.

A. Linking challenges

Some problems of publishing datasets as Linked Data are aroused when linking the dataset to other datasets. Here, we discuss some of these challenges.

1) choosing appropriate ontologies and predicates

An important issue in publishing linked data is to decide which ontologies and predicates should be used to describe the resources. The most common solution is to select ontologies based on their popularity. Some ontologies have become the de facto standard in specific domains (for instance FOAF for personal information, or Dublin Core for information about publications). Although having good knowledge about well-known ontologies related to the domain of the dataset eases this decision making, but there are two problems in this regards: first, this popularity-based approach is not effective for all cases (e.g. for domains which there is no well-known ontology), and second, there is not any automatic approach to systematically identify and evaluate candidates.

The approach used in this project is an ad-hoc one. For domains which there is a de facto standard ontology (e.g. FOAF, or Dublin Core), it is chosen. Otherwise, when there is no such ontology, a subjective semi-automatic approach is used to find the required ontology. First Swoogle semantic search engine is used to manually search for ontologies that contain the main concepts of the domain of interest. Then, for each of the top-5 resulting ontologies, a search is performed to estimate the popularity of that ontology on Linked Data space. To do so, a number of SPARQL queries are executed on the LOD SPARQL endpoint to see how many times the predicates of that ontology are used in the LOD cloud. The most common used ontology is then selected as the most appropriate one. As an example, when describing the data of professors in FUM-LD, a predicate was required to specify that a special course is taught by a professor. Using the approach described above, the predicate 'teaches' from dmcNS was selected.

[5] http://devel.patrickgmj.net/dmcNS

[6] http://dbpedia.org/property#

Considering how ontologies are selected, it is not easy to evaluate quality of published data. Also using approaches which involve manual repetitive activities and subjective judgment increases the publication cost and spent-time and decreases the accuracy and quality of the results, especially for large, dynamic and complex datasets. Therefore, one of the challenges of linked data is lack of a standard well-defined approach for choosing required ontologies and predicates.

2) creating appropriate links between data

Another challenge is related to finding appropriate links between resources from different datasets. This link discovery process requires using specific record linkage [8] and duplicate detection [3] techniques developed within the database community, as well as ontology matching [4] methods from the knowledge representation literature. This link discovery activity consists of two main steps: 1) Identifying the logic of linkage, i.e. deciding which resources under which conditions can be linked to others using which link types. 2) Searching and finding the instances of such resources and links in datasets. Silk [14] framework can be used to perform the second step automatically by a formal specification of linkage logic, but the first step is still a challenge and domain experts have to to identify the linkage points and specific rules required for finding the links.

In this project, through analyzing FUM database, and browsing related linked data sets like DBpedia and ACM, and performing some manual schema matching, some heuristics are found for logic of linkage. Based on these heuristics, a link discovery procedure is developed inside RDFizer which uses a string matching algorithm with an experimentally adjusted threshold. During experiments, a number of such algorithms, implemented in SimMetrics[7] tool, are studied and Levenshtein, JaccardSimilarity and CosineSimilarity algorithms are selected as candidate. So, 12000 pairs are compared using three algorithms with 6 different threshold values. Then the results are evaluated by members of the team. Finally for each one, four metrics of true positive, true negative, false positive, and false negative are calculated. About 7500 pairs from 12000 pairs are related to the names of persons, and others are related to the titles of papers. Result of this experimental phase is presented in the Appendix.

After analyzing these experimental results, it was decided to use Levenshtein algorithm for the string matching phase with different thresholds: value of 0.8 for matching title of papers and value of 0.9 for names of persons.

It can be concluded that the process of link discovery, and especially determining the logic of linkage require expertise, detailed understanding of the dataset at hand, as well as familiarity with external datasets and ontologies.

B. Data Challenges

1) lack of data or presence of low-quality data

One challenge in publishing a dataset as linked data is lack of required data in the original dataset. For instance in FUM-LD project, it was observed that the original database does not contain any information for publications of many professors. Even in cases that such information exists in the database, incomplete and incorrect data are entered. The reason is that the system front-end has not performed appropriate control or validation on data entered by end-users. For instance, for some papers, data about abstract or keywords, or list of coauthors does not exist in the table of papers in the database. Different types of formats are used for entering date values (e.g. date of a conference). Also, there were Persian data in columns that should contain English data, or vice versa (e.g. there are 2 columns for storing names of professors, one for Persian, and the other for English, but English column contains Persian data). In systems such as professor portals, where data is not considered as important operational data, and it is left to the end-users to freely enter their data, such problems of low-quality or missing data lead to challenge when it comes to linking resources to related ones in external datasets.

To address this challenge, it is required to precisely analyze original data and identify existing problems, and then use the data cleansing techniques or customized ad-hoc solutions to fix the problems as much as possible. For instance, it is possible to implement algorithms to convert different formats of dates to a unique format, or to move Persian values from English columns to the corresponding Persian columns. Unfortunately, such customized solutions are specific to the dataset at hand, and have low reusability in terms of publishing datasets of a different domain. In addition to such data cleansing solutions, it is possible to use linked data itself to identify appropriate values for missing data. For instance, after linking a resource of type paper from FUM-LD to its corresponding resource in DBLP, it is possible to extract names of coauthors (or other attributes, e.g. keywords) from DBLP and add them to the specification of that resource in FUM-LD.

C. Persian Language challenges

Since most data on LOD cloud is published in English, it is hard to link a Persian dataset to the related external datasets. To the best of our knowledge, there is no work in the literature discussing this problem, even for other non-English datasets. In multi-language systems where data is generated freely by ordinary end-users, it is possible that some users choose their mother tongue language while others use English for entering their data, whether for their convenience, or because of their field of activity. For instance, in the FUM database, for the engineering faculty members, data mostly contains English data, while for the theology faculty members, Persian and Arabic data is dominant. As another example, identical Persian terms exist in different English forms in the database, e.g. a single Persian name "سعید" is entered both as "saeed" and "saeid". Such problems caused by multi-lingual data, introduce challenges when searching external datasets for related resources to be linked, and decrease the quality of the published dataset.

One way of addressing such problems is to use a dictionary to identify different equivalences of a word from one language to another. For instance, in FUM-LD framework, the dictionary element provides access to different equivalences of a Persian name in English. Using this dictionary, it is possible to use all

[7] http://sourceforge.net/projects/simmetrics

equivalences of a professor name, when searching external datasets. Therefore, the probability of missing a related link because of different spelling is reduced.

D. Data and Link Maintenance

An important issue in maintaining the quality of data published as linked data is to update this data as well as the existing links between the data items. When updating the dataset, information about the time of creation and modification of data is published along the dataset. Predicates like dcterms:created and dcterms:modified can be used to store such information in voiD specification of the dataset. In order to have a successful update process, it is required to consider the type of published data, rate and frequency of data changes in adjusting the update interval.

There are different kinds of changes which require updating the dataset, but generally speaking, there are two main situations that requires updating the dataset.

1. The original dataset is changed. For instance, in case of FUM-LD project, if a new professor joines a department, new resources of types professor and paper should be added to the dataset, new internal links of type foaf:knows should be created between this professor and his colleagues (professors of the same department), new links might be available for linking these new resource to other resources in external datasets, for instance linking the new professor to a resource in ACM using owl:sameAs link.
2. A related external dataset is changed. Similar to the original dataset, external datasets might also change by introducing new resources or links. If the original dataset is linked to such an external dataset, it requires to be updated. For instance, if a new resource describing 'Computer Engineering Department of Ferdowsi University o Mashhad' is added to DBpedia, then it is a good candidate to be linked by the resource which describes the same thing in FUM-LD. If external datasets specify their last modification timestamp (e.g. in their voiD specification), then publishers of the original dataset are able to decide when to update their dataset. If an external dataset is updated monthly, all the links to this dataset should be updated monthly.

Therefore, from the point of view of the consumers, it is required that the times of creation and last modification of the dataset are specified to help them judge about the trustworthiness and validity of data. So, timestamps in four granularity levels can be used for this reason:

1. Original dataset level: it is possible to use a timestamp for the whole dataset to specify its creation and last modification date/time.
2. External dataset level: timestamps can be used for each of the external datasets that the original dataset is linked to. For instance, in FUM-LD project, it is possible to specify in voiD specification the last date of linking FUM-LD to ACM dataset. Therefore, a user who is following a link from a FUM-LD resource to related ACM resource knows when this link was created, and then can have a sense of validity of the link.
3. Resource-level: timestamps can be attached to each of the resources, to specify when it was created or modified.
4. Triple-level: at the lowest level, timestamps can be assigned to each triple, providing information about when it was created.

Based on the chosen granularity level, overhead of using timestamps varies. Also the possible update level varies.

At the topmost level, only 2 triples are required to specify the creation and last modification timestamp of the whole dataset, while at the lowest level, each triple is accompanied by one extra triple (if only last modification timestamp is used). Therefore, the lower the granularity level, the more space is used for the timestamps. If the dataset is finally published using a triple store, then from a query execution point of view, it is not a good idea to fill the triple store with too many timestamp triples that might have no use in query answering.

Using the topmost level, it is only possible to update the dataset as a whole, since the timestamps are used at the whole dataset, while using triple-level timestamps, it is possible to update triples independently. Therefore, the lower the granularity level, the more flexible the update process is.

In FUM-LD project, the second level is used, i.e. timestamps are used at external dataset level.

V. CONCLUSION AND FUTURE WORK

In this paper, some problems and challenges of publishing Persian linked data are discussed based on our experience, "FUM-LD". By analyzing the empirical results of this project, some publisher-oriented approaches are proposed to improve the quality of linked data.

Since, the main focus of this project is on publishing data of Ferdowsi University of Mashhad, we are going to improve FUM-LD framework. So, our future works include developing a comprehensive framework to publish academic linked data and proposing a data model for this framework.

VI. REFERENCES

[1] K. Alexander, R. Cyganiak, et al. "Describing Linked Datasets", Linked Data on the Web workshop (LDOW2009). Madrid, Spain.

[2] T. Berners-Lee, (2006), "Linked Data. Design Issues for the World Wide Web" http://www.w3.org/DesignIssues/LinkedData.html.

[3] A. K. Elmagarmid, P. G. Ipeirotis, et al. (2007). "Duplicate record detection: A survey." IEEE Transactions on Knowledge and Data Engineering 19(1): 1-16.

[4] J. Euzenat, P. Shvaiko (2007). "Ontology Matching." Springer, Heidelberg.

[5] Y. Raimond, C. Sutton, et al. (2008). "Automatic Interlinking of Music Datasets on the Semantic Web". In International Workshop on Linked Data on the Web (LDOW 2008). Beijing, China.

[6] SKOS, Semantic Web Deployment Working Group. SKOS Simple Knowledge Organization System Reference.http://www.w3.org/TR/swbp-skos-core-spec/,2008.

[7] N. Toupikov, J. Umbrich, et al. (April 20th, 2009). "DING! Dataset Ranking using Formal Descriptions". Linked Data on the Web workshop (LDOW2009). Madrid, Spain.

[8] W. Winkler, (2006). "Overview of Record Linkage and Current Research Directions", Bureau of the Census, Technical Report.

[9] S. Paydar, M. Kahani, et.al, "Publishing Data of Ferdowsi University of Mashhad as Linked Data", International Conference on Computational Intelligence and Software Engineering (CiSE 2010), 2010.

[10] M. Rowe, "Interlinking distributed social graphs," in Proceedings of WWW 2009 Workshop on Linked Data on the Web, 2009

[11] Y. Liu and F. Z. Schar, C,, "Towards practical rdf datasets fusion," in Workshop on Data Integration through Semantic Technology (DIST2008), ASWC2008 Bangkok, Thailand, 2008.

[12] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity ooding: A versatile graph matching algorithm," in In 18th International Conference on Data Engineering (ICDE), 2002, pp. 117-128.

[13] A. Hogan, A. Harth, A. Passant, pp. Decker, and A. Polleres, "Weaving the Pedantic Web," Proceedings of the Linked Data on the Web WWW2010 Workshop (LDOW 2010), Raleigh, North Carolina, USA: 2010.

[14] Volz, J., C. Bizer, et al. (April 20th, 2009). Silk - A Link Discovery Framework for the Web of Data. Linked Data on the Web workshop (LDOW2009). Madrid, Spain.

Appendix Results of different string matching algorithms

Algorithm	Threshold	Paper titles			Professor names		
		No. of pairs	True positive (%)	False positive (%)	No. of pairs	True positive (%)	False positive (%)
CosineSimilarity	0.6	368	0.1576	0.8424	248	0.5806	0.4194
	0.7	362	0.1547	0.8453	202	0.6089	0.3911
	0.8	354	0.1469	0.8531	194	0.6340	0.3660
	0.85	343	0.1195	0.8805	110	0.8909	0.1091
	0.9	335	0.0985	0.9015	110	0.8909	0.1091
	0.95	309	0.0324	0.9676	110	0.8909	0.1091
JaccardSimilarity	0.6	57	0.9298	0.0702	183	0.6721	0.3279
	0.7	47	0.9362	0.0638	99	0.9899	0.0101
	0.8	37	0.9189	0.0811	99	0.9899	0.0101
	0.85	18	1.0000	0.0000	99	0.9899	0.0101
	0.9	11	1.0000	0.0000	99	0.9899	0.0101
	0.95	10	1.0000	0.0000	99	0.9899	0.0101
Levenshtein	0.6	71	0.8169	0.1831	1649	0.0988	0.9012
	0.7	64	0.9063	0.0938	778	0.1838	0.8162
	0.8	57	1.0000	0.0000	385	0.3221	0.6779
	0.85	56	1.0000	0.0000	317	0.3817	0.6183
	0.9	54	1.0000	0.0000	213	0.5681	0.4319
	0.95	49	1.0000	0.0000	95	0.9895	0.0105