Locality-Sensitive Hashing for Massive String-Based Ontology Matching

Michael Cochez
Department of Mathematical Information Technology
University of Jyväskylä
P.O. Box 35 (Agora)
40014 Jyväskylä, Finland
Email: miselico@jyu.fi

***Abstract*—This paper reports initial research results related to the use of locality-sensitive hashing (LSH) for string-based matching of big ontologies. Two ways of transforming the matching problem into a LSH problem are proposed and experimental results are reported. The performed experiments show that using LSH for ontology matching could lead to a very fast matching process. The quality of the alignment achieved in these experiments is comparable to state-of-the-art matchers, but much faster. Further research is needed to find out whether the use of different metrics or specific hardware would improve the results.**

I. Introduction

When people and organizations create classifications they will, most of the time, be incompatible with each other. One way to make these classifications work together is by creating a mapping between them. This article describes a possible way to create a mapping between two large ontologies. The problem when searching for such mapping for really large ontologies is that their size renders looking at all possible combinations of concepts cumbersome. The number of candidate pairs for the ontologies used in the experiments in this paper is 8,171,287,936. If assessing the quality of each pair would take 1 ms, then one would have to wait roughly three months before all combinations have been tried. In this paper locality-sensitive hashing (LSH) will be used to prune the number of pairs which have to be compared. LSH has been used earlier in the context of ontologies, for instance by de Paula et al. [1] who used the LSH technique to group similar concepts together, mainly aiming at a faster retrieval speed. A recent article by Duan et al.[2] also uses LSH for the matching of ontologies. In the article, the authors used LSH to align ontologies with around 1000 and 300 concepts or types using instance-based matching. In this article much larger ontologies with between 66000 and 123000 concepts are considered and only label information is used as an input for the LSH process.

The article is structured as follows: In the sections II and III ontology alignment and locality-sensitive hashing are introduced. Section IV contains a short discussion about possible distance measures for ontologies. Next, section V describes two strategies for transforming the ontology matching problem into an LSH problem. The section also contains a description of the implementation which is used for the evaluation in section VI. The results are compared with state-of-the-art systems and further research directions are suggested in section VII, after which a short conclusion is drawn.

II. Ontology Alignment

According to Gruber [3], ontologies are ‘an explicit specification of a conceptualization”. If we now have two different specifications, i.e., ontologies, we might need to find out how the concepts described in the one ontology O_1 relate to the concepts described in second ontology O_2. After these relations have been found, we are able to translate instances described in the one ontology to the other ontology. This further enables us to answer queries asked using vocabulary from either ontology using data from both ontologies. A set of relations between the concepts of the two ontologies are called an *alignment*, which is formalized as follows:

Given an ontology O_{left} with matchable entities $Q_L\left(O_{left}\right)$ and an ontology O_{right} with matchable entities $Q_L\left(O_{right}\right)$, an alignment A is a set of triples (l, r, rel) with $l \in Q_L\left(O_{left}\right)$, $r \in Q_L\left(O_{right}\right)$ and $rel \in \{=, \sqsubseteq, \sqsupseteq\}$. One such triple is sometimes called a correspondence. The meaning of a triple is that the concept l is related to the concept r by relation rel, where $=$ means that the concepts are equal, $\sqsubseteq$ that l is less general as r, and $\sqsupseteq$ that l is more general as r. This definition is an instantiation of the more abstract definition given by Euzenat and Shvaiko in [4].

When assessing an alignment, one often uses a so called gold standard. This is an alignment which is considered correct and used for evaluation of other alignments. Typical measures to assess the quality of an alignment with regard to this standard are precision and recall [5]. Precision is a measure which indicates how correct the resulting mapping is, while recall measures how many mappings are missing. These measures are often unified in the so called F-measure, which is the harmonic mean of precision and recall. For the remaining of this article we will, for clarity, distinguish between the left and right ontologies with l and r concepts respectively. When ontologies are matched, it is of no importance which one to choose as left and right.[1]

[1]As will be shown in section V, one can optimize the storage space needed if the ontology with less concepts is taken as the left ontology. The amount of signatures is linear with the size of the ontologies and only the ones for the left ontology have to be stored in memory all at once. However, the signatures are much smaller as the actual ontologies and hence the gain of switching the ontologies is not significant on modern hardware. In order to show the robustness of the system, the biggest ontology is chosen as the left one for the experiments.

III. Locality Sensitive Hashing

Locality-sensitive hashing (LSH) was first introduces by Indyk et al.[6] as a method for finding approximate nearest neighbors, given a distance measure, d, and a threshold for the error, ϵ. The method first selects a number of hash functions for which the probability of a collision is high if the hashed objects are similar. If objects are dissimilar, the hash functions are very likely to hash them to separate buckets. Now, to find near-neighbors of a query point, one hashes that point with each of the hash functions and returns the elements stored in the buckets the point gets hashed to.[7]

Concrete, to apply LSH we need a family $\mathcal{H}$ of hash functions which map from a space $\mathcal{D}$ to a universe $\mathcal{U}$.

Let $d_1 < d_2$ be distances according to a distance measure d on a space $\mathcal{D}$. The family $\mathcal{H}$ is (d_1, d_2, p_1, p_2)-sensitive if for any two points $p, q \in \mathcal{D}$ and $h \in \mathcal{H}$:

- if $d(p,q) \leq d_1$ then $Pr\left[h(p) = h(q)\right] \geq p_1$
- if $d(p,q) \geq d_2$ then $Pr\left[h(p) = h(q)\right] \leq p_2$

where $p1 > p2$.

Using one function from $\mathcal{H}$ to decide whether points are likely similar is not enough since the probabilities p_1 and p_2 might be close to each other. In LSH this problem is solved using amplification. This amplification is done by creating b functions g_j, each consisting of r hash functions chosen uniformly at random from $\mathcal{H}$. The names b and r stand for *bands* and *rows*. If one puts all outcomes of the hash functions of all g_j in a two-dimensional table, it can be split in b bands each consisting of r rows. A function g_j only maps points p and q to the same bucket if all hash functions it is build from hash the points to the same buckets. If for any j, the function g_j maps p and q to the same bucket, p and q are considered close. Notice that the amplification also creates itself a locality sensitive family which is $\left(d_1, d_2, 1 - \left(1 - {p_1}^r\right)^b, 1 - \left(1 - {p_2}^r\right)^b\right)$ sensitive.

A very accessible introduction to LSH can be found from [8]. Some notations used in this section are borrowed from that book.

IV. Distance Measures For Similar Concepts

Several metrics introduced for concept distance work only when both concepts reside in the same ontology. Examples of these include metrics which measure the minimal path between concepts, shortest path in a is-a hierarchy, and depth with respect to a common subsumer. [9] For measuring distance between concepts from different ontologies, we can use information which is related to the concepts in their respective ontologies. The ways of comparing concepts found in the literature can be classified in three groups [4], namely matching using information available in labels and names of the concepts, using structural information or using information from instances defined using the ontology. These different ways can be used either in isolation or combined. In the scope of this article, only information available in labels will be used (see section V-A).

V. LSH for Ontology Alignment

When we regard ontology alignment as a nearest-neighbor problem, we can consider all concepts of the left ontology as being the collection of points from which we query. Subsequently, each point of the right ontology can then be seen as a query point for which we try to find a close match. When we query for nearest neighbors for each concept c_r from the right ontology, we will get a set of concepts from the left ontology which are likely similar to c_r. This likelihood is depending on the distance metric associate with the locality-sensitive family $\mathcal{H}$ and the values of r and b.

A. Distance Metric

For our experiment, the Jaccard similarity for sets was used. Using this metric, the similarity between two sets $A, B \subseteq S$ is measured as $sim(A,B) = \frac{|A \cap B|}{|A \cup B|}$. The corresponding Jaccard distance measure is $d(A,B) = 1 - sim(A,B)$.

There are two aspects to be considered before this metric can be used for LSH. Firstly, we need to find a family of locality-sensitive functions with respect to this distance. Secondly, we start from concepts in an ontology and hence we need to transform the concepts into sets between which the Jaccard similarity can be measured.

The first task has been solved by Broder [10] using the LSH family called min-hash. First, a random permutation π of the universe S is chosen. Then, the hash function is defined as $h_\pi(A) = \min\left\{\pi(a) \,|\, a \in A\right\}$. Using this definition, the probability of a collision becomes $Pr_\pi\left[h_\pi(A) = h_\pi(B)\right] = sim(A,B)$. Therefore, we end up with a $(d_1, d_2, 1 - d_1, 1 - d_2)$ sensitive family of hash functions.

For the second task a strategy inspired by Duan et al.[2] is used. The authors propose to use a document consisting of the concatenation of all the values of the *rdfs:label* property of all instances of the concept. This document is then processed and all terms are extracted. The distance is then defined as the Jaccard distance between the term sets of concepts.

In this article, however, no instance information is used. Hence, there is a need to develop novel strategies for measuring the distance between concepts only based on their label information. Two strategies were developed and used in the experiments below.

The first strategy involves the computation of the so-called n-grams of each label of each class. The set representing a concept is then taken as the union of all n-grams of the lowercased labels of the class. Now, the distance between two classes is defined as the Jaccard distance between these shingle-sets and hence min-hashing and LSH can be used.

The second strategy does not unite the labels of the class. Instead, for each class multiple sets are created, one for each label of the class. The labels are converted into sets as follows. First, the label is lowercased and split on whitespace or other punctuation marks. Next, if this operation created substrings of length 1, they will be merged back to the substring which come before it. All substrings of length 1 which could not be merged back, are removed. Finally, the concept is represented in the system as many times as it has labels. Each representation uses a set of the generated substrings to represent the class.

B. Implementation

The implementation used for our evaluation is programmed in Java and heavily uses parallelism to speed up the computation. From the description of the LSH algorithm, it can be noticed that the hashing of the concepts happens independent of each other. Therefore, they can be computed in parallel using a multi-processor system. Also multiple queries for nearest neighbors for the concepts of the right ontology can happen at the same time.

Inspired by [8], normal uniform hashing was used several times to speed up computations or save on storage space. Firstly, shingles are not stored as sequences of characters, but hashed down to a lower dimensional space so that they would occupy less space and compare faster. Secondly, the creating of permutations of S is avoided (which would be computationally unacceptably expensive). So, instead of creating a real permutation random (non locality-sensitive) hash functions are used which map each original index to a target index. The last place were normal random hashing is used to speed up computations is during the query phase. The elements of the signature of a concept in the same band are hashed to a single value. This hashing is performed using Rabin fingerprints as described by Broder [11].

VI. Evaluation

The proposed strategies are evaluated by conducting various experiments. The conducted experiments have several parameters. There are the datasets which are described in section VI-A and then there are the parameters used for the LSH algorithm. These parameters are the number of rows r and the number of bands b. For the first strategy, there is also the shingle size.

There are several things which are interesting to measure. First, it is possible to see how many distinct shingles the concepts of the ontologies contain and the time needed to do the pre-processing. Then there is the time needed to calculate the min-hash signatures from the shingle sets. And finally the time needed to query the left signatures for concepts similar to the right signatures. After the experiment, the proposed mapping can be obtained and its quality evaluated. This quality can be evaluated by comparing its performance to (i) a random algorithm and (ii) a hypothetical perfect one, i.e., a gold standard.

For the random algorithm, the same number of pairs which the proposed algorithm produces are randomly selected and all of them are considered positives. Then the precision and the recall of this algorithm are calculated. The performance of the random algorithm can be predicted from the number of positives n, the size of the left ontology l, the size of the right ontology r, and the number of alignments in the reference alignment a. The expected values for the performance of the random algorithm can be found in table I. It is clear from the formulas that if $l * r$ is large compared to a, the random algorithm will have a very bad performance. Since our ontologies are large and our alignment relatively small, the random approach will produce results for which the precision and recall are very close to zero. Therefore there will be no literal comparison to this algorithm, but it should be noted that if a correct correspondence is obtained, it is very unlikely that it was created by coincidence.

The perfect algorithm is hypothetical and produces only true positives and hence has a precision of 1 and a recall of 1. In practice, this algorithm just uses the reference alignment to generate pairs. Comparing with the reference alignment gives an evaluation for the whole procedure.

A. Datasets

The datasets used in the evaluation are taken from the *Large Biomed Track* of the Ontology Alignment Evaluation initiative[2]. The first dataset is the FMA ontology[3], which contains 78,989 classes, the second is the NCI ontology[4] containing 66,724 classes, and finally a fragment of the SNOMED ontology[5] which contains 122,464 classes. This choice of datasets is mainly to make comparison of matching systems possible. The FMA ontology only contains classes and non-hierarchical datatype properties, i.e., no object or datatype properties nor instances. The NCI ontology contains classes, non-hierarchical datatype and hierarchical object properties. The classes of all ontologies are structured in a tree using *owl:SubClassOf* relations. As a gold standard, the UMLS-based reference alignments as prepared for OAEI 2013[6] are used. Only the equal correspondences are retained and all confidence levels are considered one.

B. Experiments

The first two experiments are concerned with the first proposed strategy. Since the strategy has a n-gram size, an attempt is made to find out how the n-gram size affects the quality and speed of the algorithm. Therefore, a fixed number of bands (20) and rows (24) is chosen and the algorithm is run for n between 1 and 20. Then graphs are created which relate the n-gram size with the different timings and the result quality indicators.

In the second experiment, a fixed shingle size (6) is used and different values for the number of bands b and rows r are tried. To keep these comparable, it is ensured that $b * r = 480$. The number 480 is chosen because it has many divisors and it complies with the 'several hundred permutations' as proposed in [8]. Also these experiments are graphed and discussed.

The third experiment evaluates the second proposed strategy. This strategy does only have the bands and rows parameters and hence only one series of experiments is conducted. The experiment is essentially the same as the second one.

To keep these experiments comparable, the NCI ontology and SNOMED fragment are used in all experiments. The experiments are executed on a OpenJDK 64-Bit Server VM. The Java VM runs on hardware with two Intel Xeon E5-2670 processors (totaling 16 multi-threaded cores).

[2]http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2013/

[3]http://sig.biostr.washington.edu/projects/fm/

[4]http://www.obofoundry.org/cgi-bin/detail.cgi?id=ncithesaurus

[5]http://www.ihtsdo.org/index.php?id=545

[6]http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2013/oaei2013_umls_reference.html

TABLE I. EXPECTED VALUES FOR THE RANDOM ALGORITHM. CONCRETE VALUES ARE CALCULATED USING VALUES FROM SECTION VI-A FOR THE NCI AND FMA ONTOLOGIES, USING $n = 10000$.

	TRUE	FALSE
Positive	$\frac{n}{l*r} * a = 0.00548$	$n * (1 - (\frac{a}{l*r})) = 9999.99451$
Negative	$(l * r - a) * (1 - \frac{n}{l*r}) = 5270449146$	$a * (1 - \frac{n}{l*r}) = 2889.99451$
Precision $\approxeq 5.48 * 10^{-7}$		Recall $\approxeq 1.90 * 10^{-6}$

C. Results

The results of the first experiments are shown in the charts in figs. 1 to 3. The first chart shows, in function of the n-gram size, the number of distinct shingles created from the labels, the number of unique shingle concept pairs and the time needed to calculate the shingles. From the graph it can be noted that the longer the n-grams are, the faster they are found. This is as expected since, given a fixed string, the number of n-grams is lower for a larger n. Further, the longer the shingles are, the more distinct shingles are found. This is again not surprising since there are more possible shingles when they can be longer. Finally, the Shingle to concept mapping indicates that there are less shingles found overall.

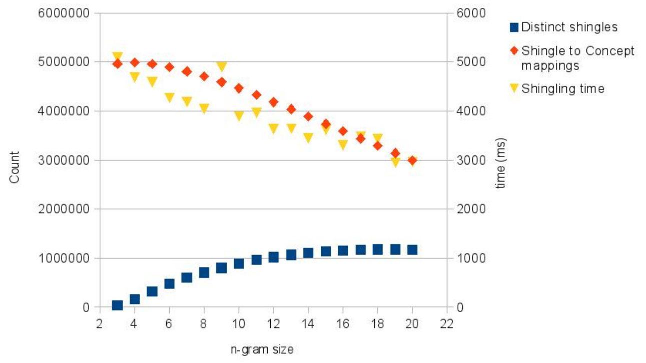

Fig. 1. Number of distinct shingles, shingle to concept mappings and shingling time in function of the n-gram size

The second chart shows the time needed for the different parts of the algorithm where the calculation of the signature of the query points is included in the signature calculation time. The signature calculation is very fast, and very similar for the different n-gram sizes. There were a couple of calculations which took slightly longer, but a clear pattern was not found. The pre-processing is slower for smaller n-gram sizes since there are more shingles to be calculated. The first two measurements for nearest-neighbor querying are clearly not correct. They will be discussed further in the next paragraph. Besides these two, the calculation of the nearest neighbors is faster when there are larger shingles. This can be explained by the fact that there are less shingle to concept mappings.

From the third and last chart it can be seen that, all in all, the size of the shingles does not affect the quality much. Both precision and recall are more or less constant. When looking carefully, it is seen that recall and hence the F-measure decrease slightly. This is explained by the fact that when the shingles are short, there will be a larger fraction of the true positive pairs which gets tested by coincidence. In both the second and third figure, it was clear that n-grams of size

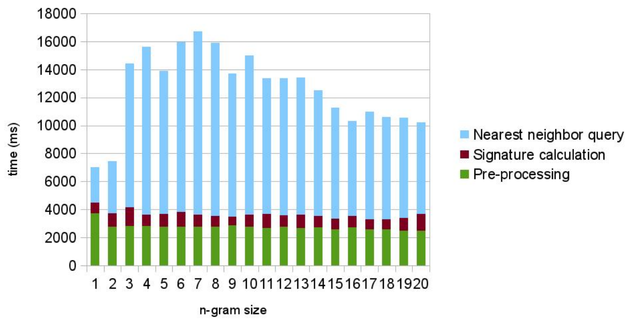

Fig. 2. The time for the various parts of the LSH process in function of the n-gram size.

one and two did not perform normal. The reason for this are the Rabin fingerprints, which do not work well for too small inputs.

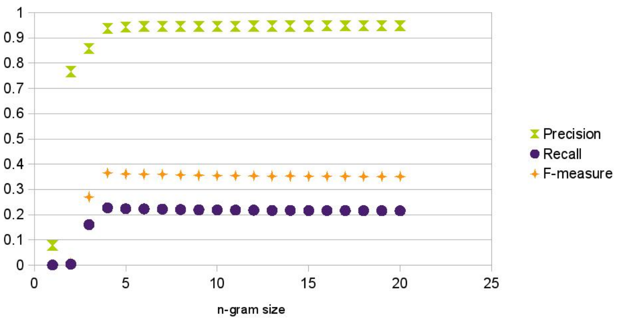

Fig. 3. The precision, recall and F-measure in function of the n-gram size.

For the second experiment, with fixed shingle size, the results are shown in figs. 4 to 6. The pre-processing is not included in the graph since it happens before the number of rows and bands are chosen and is hence not affected by this choice. The signature calculation seems independent of the number of bands. This is as expected since the amount of signatures to be calculated is dependent on the product of r and b and hence constant. There does not seem to be a clear patter in the behavior of the nearest neighbor calculation time. In future research the behavior of this curve could be further investigated.

The next figure (fig. 5) shows the fraction of true positive and false positive results. As can be seen from the chart, the more bands in use, and hence the fewer rows per band, the more false positives are generated. The reason for this behavior is that reducing the amount of rows per band causes more pairs to be generated. From a theoretical point of view, one can see

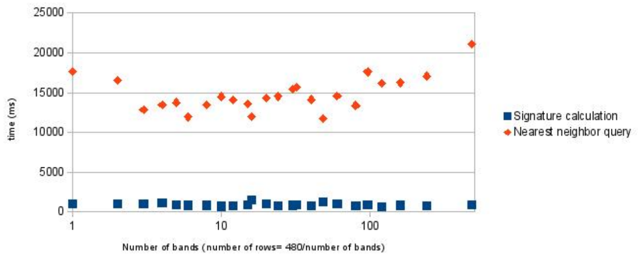

Fig. 4. The time for signature calculation and nearest neighbor querying in function of the number of rows per band. Note the logarithmic scale.

that if there is a chance p to find a match within 1 row, then there will be a p^r chance to find a match in r rows. Hence, if there are x rows, the chance will be p^x, while with $x/2$ rows this will be $p^{x/2} = \sqrt{p^x}$.

The last graph of this experiment (fig. 6) indicates the precision, recall and F-measure for each band size. This graph shows that the precision goes down when less rows are required to agree, i.e., the band size is smaller. It, however, also indicates that overall more correct pairs are found, which is seen from the bigger recall for smaller bands. These observations correspond to the expectation described in the paragraph above.

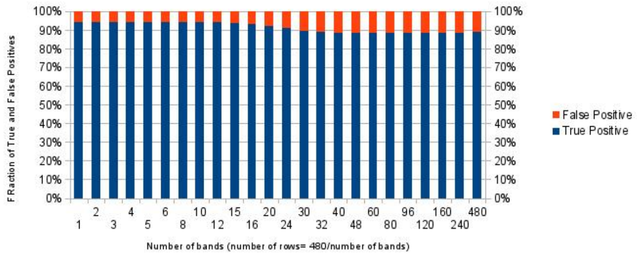

Fig. 5. The fractions of true and false positives in function of the number of rows per band.

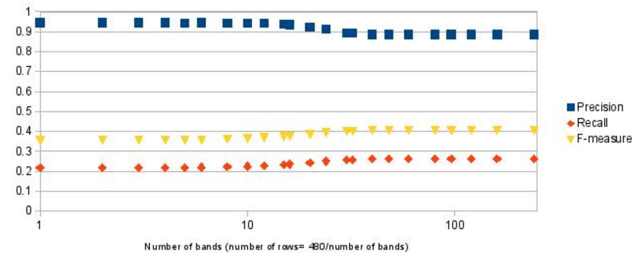

Fig. 6. The precision, recall and F-measure quality indicators in function of the number of rows per band.

The third experiment uses the second strategy where each concept is represented in the system as many times as it has labels. In this experiment the ontologies, with 66,724 and 122,464 classes, are expanded to 122,508 and 179,238 representations respectively. The results of this approach seem promising and are illustrated in figs. 7 to 9. As in the previous experiment, the signature calculation (fig. 7) is fairly constant. When comparing the signature calculation time between this and the previous experiment, it can be noted that calculating the signatures is only very slightly slower. The average speed for the previous experiment was 929ms as compared to an average of 1135 ms in this experiment. As discussed in the previous experiment, there is no clear pattern in the behavior of the nearest neighbor query time. The clearly outlying measurement for the one big band of 480 rows is caused by the fact that the current implementation is not able to parallelize the neighbor querying.

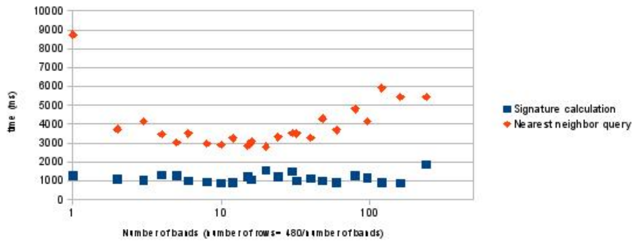

Fig. 7. The time for signature calculation and nearest neighbor querying in function of the number of rows per band. Note the logarithmic scale.

fig. 8 shows the fraction of true and false positives. When comparing to fig. 5 from the previous experiment, there are relatively more false positives. From fig. 9 it can be seen that the false positives are the price which one has to pay to get a much higher recall as in the previous experiment. Because of the larger recall and the only marginally worse precision a much higher value for the F-measure is obtained.

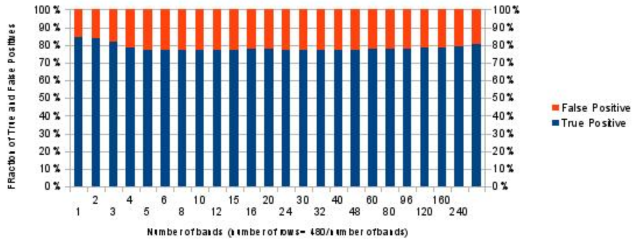

Fig. 8. The fractions of true and false positives in function of the number of rows per band.

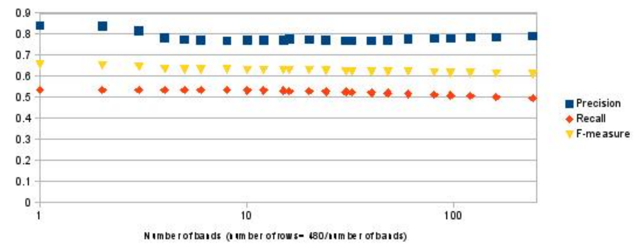

Fig. 9. The precision, recall and F-measure quality indicators in function of the number of rows per band.

VII. Discussion and Future Work

The information from the ontology used by this system is fairly limited. Only the label information is retained and all relational information about the classes is ignored. Further, information about object or datatype properties is not incorporated, nor are instances used to enrich the matching. Hence, one can not expect that the system could perform comparably to other, more advanced, matching systems. However, when comparing our findings to the systems used in the 2013 Large BioMed Track of the OAEI[7], it seems that the second strategy performs fairly well. The best result was obtained when using 1 band with 480 rows. The concrete result was a precision of 0.842 and a recall of 0.535 leading to an F-measure of 0.654. When comparing with ServOMap, the best system which performed this experiment in the evaluation, we find an F-measure 10% higher as our result. This is significant, moreover because also the confidence levels are taken into account for the ServOMap system. The price for this improvement is a runtime of 6320 seconds, roughly 300 times longer than our system, which gave a result after 21.5 seconds. When looking at the fastest matching system, the LogMapLt system is the fastest in the evaluation with a runtime of that system is 132 second or about 6 times slower the proposed system. Also, the LogMapLt system scores somewhat worse when comparing to the second strategy proposed in this paper, but again the LogMapLt figures include the confidence levels.

Obviously, it is not fair to pick the best result for our system and compare it with one of the results of the other systems. Therefore, the same strategy was applied to the two other combinations which were used in the evaluation, namely a matching of the SNOMED with the FMA ontology and the NCI with the FMA ontology. Further, also smaller subsets of these ontologies were tested to obtain results which can be compared with the top systems in the evaluation[8]. The results of these experiments are summarized in table II.

When comparing the summary of the results for precision, recall and F-measure with those of the top systems in the evaluation, the proposed system ranks at the lower end. Concretely, in terms of precision, recall and F-measure the system ranks right after LogMapLt [12] whose results are also included in table II. When the speed of matching is observed, however, the proposed system is much faster than the other systems. The LogMapLt system was the fastest and completed within 371 seconds. The proposed system needs 54.3 seconds to complete the six tasks and is hence 6.8 times faster. The table also includes the results for the best system in the competition, the YAM++ system [13], which scores much better with regard to precision, recall and hence F-measure. However, to obtain these results the YAM++ system needs 2066 seconds, which is roughly 40 times more than the proposed system.

When a matcher produces a mapping between two ontologies, the mapping might be incoherent, i.e., the mapping combined with the original ontologies entails axioms that do not follow from the ontologies or the mapping. Several systems, including the YAM++ system, perform a mapping repair operation which attempts to remove a small set of mappings in order to make the resulting mapping coherent. The proposed system does not perform any checking on the results and therefore the coherence of a proposed mapping

[7]http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2013/results2013_snomed2nci.html

[8]http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2013/results2013_top.html

TABLE II. RESULTS OF APPLYING THE SECOND STRATEGY USING 480 ROWS IN 1 BAND ON PAIRS OF DATASETS USED IN THE 2013 LARGE BIOMED TRACK OF THE OAEI.

	Time (s)	Precision	Recall	F-measure
Task 1: FMA-NCI small fragments	1.6	0.937	0.802	0.864
Task 2: FMA-NCI whole ontologies	12.6	0.641	0.800	0.712
Task 3: FMA-SNOMED small fragments	2.0	0.954	0.187	0.313
Task 4: FMA whole ontology with SNOMED large fragment	11.4	0.893	0.187	0.310
Task 5: SNOMED-NCI small fragments	5.2	0.935	0.535	0.680
Task 6: NCI whole ontology with SNOMED large fragment	21.5	0.842	0.535	0.654
Summary	54.3	0.867	0.508	0.589
Summary for the LogMapLt system	371.	0.874	0.517	0.598
Summary for the YAM++ system	2066.	0.942	0.728	0.817

notes: The times measured do not include the class loading of the Java virtual machine which occurs once only. When an experiment is repeated, its run-time is roughly 0.8 seconds shorter. The shorter time is reported.
The summary of the time is the sum of the run-times of the experiments, other reported figures are averages. This summarization enables a comparison with the top systems at the OAEI 2013 Large BioMed Track which is also the source for the results of the other systems[8]. Note that the results for LogMapLt and YAM++ account for confidence levels while the proposed system does not.

will be dependent only on the similarity of the labels of the ontologies and the number of violating mappings this causes.

The results of the proposed system are promising, especially using the second strategy the proposed system is able to find a mapping of reasonable quality very fast. Further experimenting should point out whether the strategy is useful for a broader set of ontology matching problems. The continuing fight between efficiency versus effectiveness been broadly discussed by Ermolayev et al. [14]. What this means in practice is that if one wants to find solutions faster, one will have to give up quality, and if one wants better results, one will have to wait for them. Obviously, it would be possible to build a hybrid system which would first provide a very fast result and then present corrections after some time.

There are still many aspect of using locality-sensitive hashing for ontology matching which should be researched. Firstly, one could try to fit different metrics and their accompanying locality-sensitive families to the ontology matching problem. This could be the cosine distance and the random hyperplane hash as was tried in [2] for instance-based matching, but also other distance metrics and ways of fitting them to the problem could be tried.

Secondly, there is the issue of finding good parameters for the signature calculation and its banding. We refrain from making any conclusions about good parameters since our sample is too small. More alignment problems should be tried and perhaps a set of good parameters can be found. Another approach could be the use of LSH forests as proposed in [15]. The use of LSH forests removes the need of specifying the parameters, but seems to be more difficult to parallelize.

Thirdly, it would be interesting to see how the system which was used in this paper would work on different hardware. Since the computation can be performed in parallel, it might be possible to create a version which can be executed on GPUs which would enable a tremendous parallelization for a reasonable price.

Lastly, it seems reasonable and possible to use multiple LSH processes at the same time and only retain those pairs which are selected by a certain number of processes. These processes can each use a different LSH strategy and work completely in parallel. It is also possible to create a weighted sum of the processes and if the sum reaches a certain threshold, the pair will be proposed as a solution. This weighted sum would be subject to an on-line optimization process which could perhaps be solved using some form of evolutionary computing.

VIII. Conclusions

In this paper an effort was made to solve the problem of matching two ontologies by applying locality-sensitive hashing. Overall, a string-based alignment was performed where label information of the classes was used to create the alignment. In a first strategy, this label information was shingled and each of the shingle sets was merged to form one big set representing the class. A second strategy represented each class multiple times in the system, each time with a set containing tokens from a label of the class. These representations were then used as an input to min-wise hashing. This hashing results in signatures which were in turn combined to form an amplified locality-sensitive family of functions. Finally, the one ontology was used as pool of potentially close concepts, and each concept of the other ontology was queried from it. The collection of close matches were reported as correspondences of the matching task.

Three experiments were conducted. In the first one the first strategy was used and different shingle sizes were tried, while keeping the parameters for the amplification constant. The second experiment, also using the first strategy, varies the amplification parameters and keeps the shingle size constant instead. In the final experiment, the second strategy was used in what was further a copy of the second experiment. The results of these experiments were promising. We can draw the conclusion that it is feasible to perform a matching using LSH with a relatively high speed. The second strategy seemed to work better than the first one and its quality was reasonable, often on par with state-of-the-art matching systems. Systems which produce better results are always slower, sometimes even very much so.

This is still initial research and there are still many aspects of this approach which should be investigated, like the use of other metrics, the selection of parameters, using other hardware solutions, or using multiple instances of an LSH matching process at the same time. Further, the second strategy should be tried on more ontology matching problems to prove its general applicability.

Acknowledgments

The author would like to thank the department of Mathematical Information Technology of the University of Jyväskylä for financially supporting this research. This research is also in part financed by the Need4Speed SHOK organized by the Strategic Centre for Science, Technology and Innovation in the Field of ICT (Digile Oy) and financially supported by the Finnish Funding Agency for Technology and Innovation (TEKES). The author would further like to thank the Steeri Oy for supporting the Fast Stakeholder Issue Finder business case and the members of the Industrial Ontologies Group (IOG) of the university of Jyväskylä for their support in the research. Further, it has to be mentioned that the implementation of the software was greatly simplified by the Guava library by Google and the Rabin hash library by Ian Brandt.

References

[1] L. B. de Paula, R. S. Villaça, and M. F. Magalhaes, "A locality sensitive hashing approach for conceptual classification," in *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*. IEEE, 2010, pp. 408–413.

[2] S. Duan, A. Fokoue, O. Hassanzadeh, A. Kementsietsidis, K. Srinivas, and M. J. Ward, "Instance-based matching of large ontologies using locality-sensitive hashing," in *The Semantic Web–ISWC 2012*. Springer, 2012, pp. 49–64.

[3] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?" *International Journal of Human-Computer Studies*, vol. 43, no. 5–6, pp. 907 – 928, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1071581985710816

[4] J. Euzenat, P. Shvaiko *et al.*, *Ontology matching*. Springer, 2007, vol. 18.

[5] H.-H. Do, S. Melnik, and E. Rahm, "Comparison of schema matching evaluations," in *Web, Web-Services, and Database Systems*, ser. Lecture Notes in Computer Science, A. Chaudhri, M. Jeckle, E. Rahm, and R. Unland, Eds. Springer Berlin Heidelberg, 2003, vol. 2593, pp. 221–237. [Online]. Available: http://dx.doi.org/10.1007/3-540-36560-5_17

[6] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 604–613.

[7] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Commun. ACM*, vol. 51, no. 1, pp. 117–122, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327494

[8] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2012, ch. 3. Finding Similar Items, pp. 71–128. [Online]. Available: http://infolab.stanford.edu/~ullman/mmds.html

[9] V. Cordı, P. Lombardi, M. Martelli, and V. Mascardi, "An ontology-based similarity between sets of concepts," *Proceedings of WOA, Italy*, pp. 16–21, 2005.

[10] A. Z. Broder, "On the resemblance and containment of documents," in *Compression and Complexity of Sequences 1997. Proceedings*. IEEE, 1997, pp. 21–29.

[11] A. Broder, "Some applications of rabin's fingerprinting method," in *Sequences II*, R. Capocelli, A. Santis, and U. Vaccaro, Eds. Springer New York, 1993, pp. 143–152. [Online]. Available: http://dx.doi.org/10.1007/978-1-4613-9323-8_11

[12] E. Jiménez-Ruiz, B. C. Grau, and I. Horrocks, "Logmap and logmaplt results for oaei 2013," in *Proceedings of the 8th International Workshop on Ontology Matching*, 2013, pp. 131–138.

[13] D. Ngo and Z. Bellahsene, "Yam++ results for oaei 2013," in *Proceedings of the 8th International Workshop on Ontology Matching*, 2013, pp. 211–218.

[14] V. Ermolayev, R. Akerkar, V. Terziyan, and M. Cochez, *Big Data Computing*. Taylor & Francis group - Chapman and Hall/CRC, 2014, ch. Towards Evolving Knowledge Ecosystems for Big Data Understanding.

[15] M. Bawa, T. Condie, and P. Ganesan, "Lsh forest: self-tuning indexes for similarity search," in *Proceedings of the 14th international conference on World Wide Web*. ACM, 2005, pp. 651–660.