# Title: Freshworks Invention Disclosure Form (IDF)

Instructions : Fill each blank with the requested information or enter NONE as appropriate. Where space on the form is inadequate, you can use links (documents) that points to more details. Please submit the filled IDF by filing a ticket at legal.freshservice.com

(You can replicate this form in confluence, fill it up and export as PDF and attach it to the ticket that you file in legal.freshservice.com, see solutions article <https://legal.freshservice.com/solution/categories/3000052696/folders/3000082583>)

|  |
| --- |
| PART I BASIC INFORMATION |

### 1.a DESCRIPTIVE INVENTION TITLE

|  |
| --- |
| Integrated system for entity deduplication |

### 1.b INVENTOR(S) - list yourself and any colleagues who worked on the idea with you

|  |  |
| --- | --- |
| **Name** | **email address** |
| Bharathi Balasubramaniam | [bharathi.balasubramaniam@freshworks.com](mailto:bharathi.balasubramaniam@freshworks.com) |
| Srivatsa Narasimha | [srivatsa.narasimha@freshworks.com](mailto:srivatsa.narasimha@freshworks.com) |
| Suvrat Hiran | [suvrat.hiran@freshworks.com](mailto:suvrat.hiran@freshworks.com) |
| Swaminathan Padmanabhan | [swaminathan.padmanabhan@freshworks.com](mailto:swaminathan.padmanabhan@freshworks.com) |

### 1.c INVENTOR INFO - Are any inventors located outside of India , are any of them not employed Full Time by Freshworks ?

|  |
| --- |
| No |

### 1.d PRODUCTS/TIMELINE - What products will incorporate your idea and when will these products be released ?

Freshsales - Lead/Contact Deduplication

First week of May 2019

### 1.e PRIOR PUBLICATION - Identify any publishing or journals that you referred to during formation of ideas

![](data:text/html;base64,)

|  |
| --- |
| PART II DISCLOSURE OF INVENTION |

### 2.a PURPOSE State the problem / challenge addressed by your idea.

A distributed, horizontally scalable system which can find duplicate records of entities, wherein an entity can be one of ‘person’, ‘company’, ‘institute’ or such similar abstractions, in real-time based on patterns identified from training examples. This methodology shrinks the search space required to be examined for identifying duplicates. This methodology achieves search space compression by building a corpus of tokens from an entity database. Each such token acts as an identifier or an index for a set of entity records in the database. Given an entity record, these tokens help to identify probable duplicate records by performing a significantly reduced number of comparisons. The comparison between two entity records is performed by using a machine learning model, wherein the model is pre-trained to identify duplicate entity records. A graph of duplicates is then formed wherein each vertex is an entity record and the undirected edges represent the probability of two records being duplicates of each other. These duplicates can be pulled with O(1) search time.

### 2.b PRIOR EXISTING TECHNOLOGY - Describe the previous old methods used by others and why those methods fell short. Also describe the technologies that will be integral to your solution that we do not intend to claim as an invention. This is a good place to put background that will facilitate understanding section 2c.

In most of the old methods people have seen finding duplicates in database and building an engineering system to find them in isolation. We are proposing an integrated system which can search and deduplicate records in real time as well as run distributed jobs to find and store duplicates for existing records in database. Our invention also extends to find duplicates for records related to database of people, companies, institutions and similar entities. Most other inventions talk too broadly in deduplicating databases. Our methodology ensures that our search space for probable duplicates is not too wide making it too slow a process. Our way of storing the duplicate records information lets us pull and update duplicates quickly.

Prior work

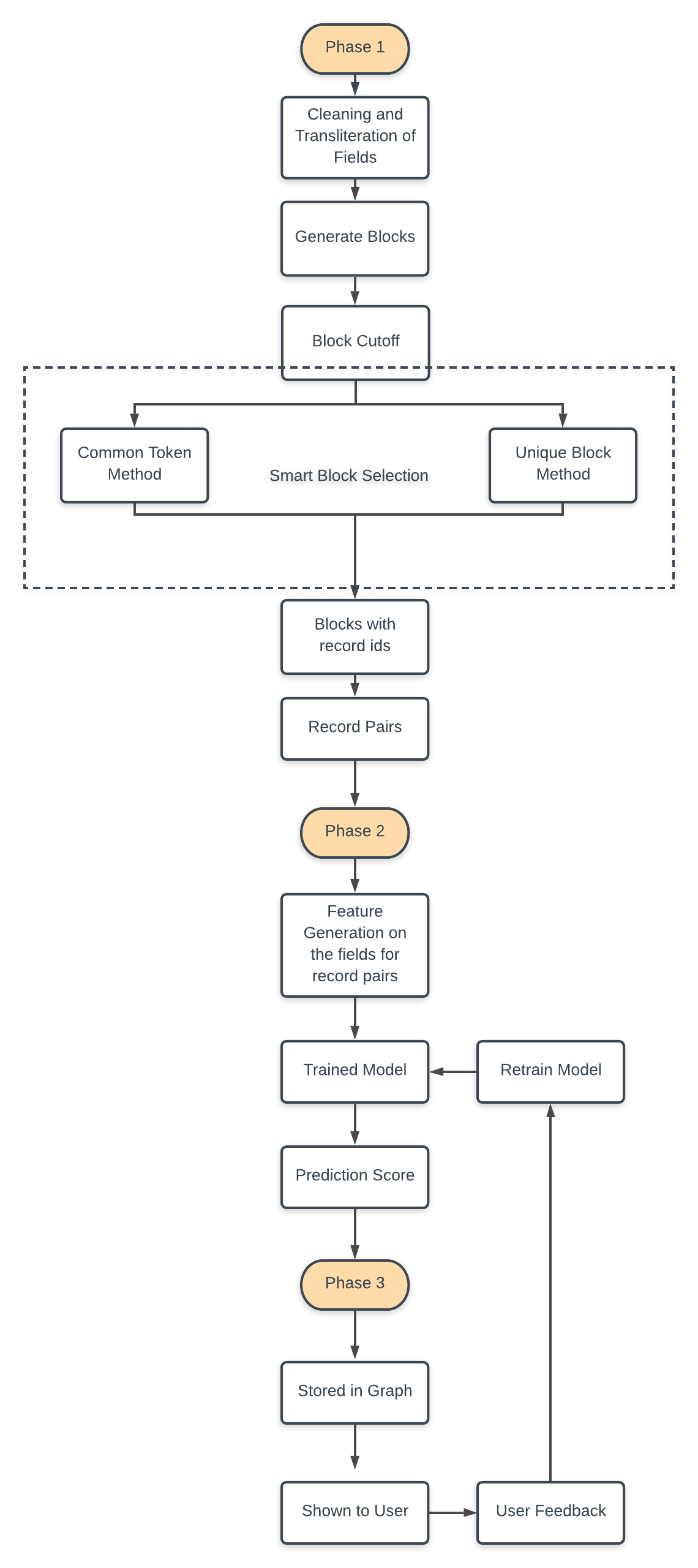
1. [Duplicate data elimination system](http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=45&f=G&l=50&co1=AND&d=PTXT&s1=%22record+linkage%22&s2=%22duplicate+records%22&OS=%22record+linkage%22+AND+%22duplicate+records%22&RS=%22record+linkage%22+AND+%22duplicate+records%22) - This patent talks about storing duplicate in a graph but lacks realtime duplicate detection capability.
2. [Reducing comparisons for token-based entity resolution](http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=3&f=G&l=50&co1=AND&d=PTXT&s1=%22record+linkage%22&s2=%22duplicate+records%22&OS=%22record+linkage%22+AND+%22duplicate+records%22&RS=%22record+linkage%22+AND+%22duplicate+records%22) - This patent is isolated for only searching records with common taken and give score based on importance of token. They do not talk about a system which can do deduplication.  They also do not talk about using an ML model for performing record comparisons.

1. [Probabilistic record linkage model derived from training data](http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=49&f=G&l=50&co1=AND&d=PTXT&s1=%22record+linkage%22&s2=%22duplicate+records%22&OS=%22record+linkage%22+AND+%22duplicate+records%22&RS=%22record+linkage%22+AND+%22duplicate+records%22) - They have used specific statistical model (maximum entropy). Our model uses random forest and feature used by us helps deduplicating entities (people, companies, etc).

Some unique features

1. Using a Machine learning model to identify duplicates
2. The process for training and validating the performance of such an ML model
   1. Including the different data sources used for training
3. Ability to generate a confidence score for duplicate records
   1. Wherein such a confidence score can be used to define the desired tolerance level for duplicate detection
4. A system that uses real-time event processing to keep the duplicates graph up-to-date at all times

### 2.c Description of invention (use flow charts or other pics to explain)



**Phase 1**

1. **Cleaning and transliteration of fields** - cleaning record fields and transliterating them if applicable to have accurate comparison for deduplication
2. **Generate Blocks** - group of records based on some tokens present in the fields to decrease the search space for finding duplicates
3. **Block Cutoff** - to remove large sized blocks which are prone to be based on tokens which are common.
4. **Smart Block Selection** - to obtain such blocks which has high possibility of genuine duplicate record pairs using an intelligent, comprehensive block selection method
5. **Blocks with record ids** - we finally have the desired blocks with the record ids.
6. **Record pairs** - create record pairs out of these blocks

**Phase 2**

1. **Feature Generation on the fields for record pairs** - calculate the features based on the fields of the incoming pair of records
2. **Trained Model** - input the features into the trained model
3. **Prediction score** - the probability score depicting whether the two records are duplicates are not is obtained from the model

**Phase** **3**

1. **Stored in Graph** - The prediction scores are stored as links between two vertices in a graph which is an efficient way of storing relationships and provides for a faster way to extract duplicates for a record.
2. **Shown to User** - This is on the product UI. Once, the user clicks on the duplicate record tab, the duplicates extracted from the graph are suggested to the user.
3. **User Feedback** - The user has choice to tag some suggestions as non duplicates. There is also a mechanism which takes in those pairs which are merged by the users as duplicates. This feedback dataset is used to retrain the model.

## **Details**

**Phase 1**

1. **Cleaning and Transliteration of fields** - Punctuations, numbers and salutations are removed and the strings are lowercased.  Stop words like company, pvt, ltd, corp, enterprise and llc, common domains in emails like [aol.net](http://aol.net), [google.com](http://google.com) and [yahoo.com](http://yahoo.com), common emails like sales, support and admin are removed. The next step is called transliteration. Transliteration is the process of transferring a word from the alphabet of one language to another. (<https://www.vocabulary.com/dictionary/transliteration>). Specifically, we detect non english languages by the presence of unicode characters in the string and transliterate them to English. We have a 1-to-1 mapping for all non English latin and cryllic characters which transliterate them to English. The reason we do this, is to find similarities between a non English record and English record which has the same name. For example, акмарал тулегенова is transliterated to Akmaral Tulegenova and hence the model will be able to figure out they are depicting the same person.

1. **Blocking** : Scaling deduplication for millions of records is challenging. We need to search for duplicity between each pair of records in a database to identify similar records. Now let's say there are 10000 records in the database, we would have to search among 49,995,000 unique pairs. This increases the computational cost and time consumption drastically. Imagine a larger database with millions of records ! Enter blocking - simply put, blocking helps us to look at only certain pairs and not all of them in a smart way. The assumption behind this method is that any two duplicate records will have at least one common value between them. If we can create groups of records (blocks) based on these common values, we can then compare only those records which are inside the blocks. If the blocks are built well, then it helps in bringing down the comparison time by a big margin. It would also help in solving the problem of non duplicates explained earlier as now, these pairs are from the same block and hence will be somewhat similar but still are not duplicates. We use a blocking approach which helps us to look at only certain pairs and not all of them in a smart way. The assumption behind this method is that any two duplicate records will have at least one common value between them. If we can create groups of records (blocks) based on these common values, we can then compare only those records which are inside the blocks. If the blocks are built well, then it helps in bringing down the comparison time by a big margin. It would also help in solving the problem of non duplicates explained earlier as now, these pairs are from the same block and hence will be somewhat similar but still are not duplicates.

**Tokenization** - The simplest way to create a block is by tokenizing a records and grouping all records belonging to a token. There are different ways of creating a token, based on the fields of the records like name, phone numbers or company names. Tokens can also be created based on parts of every word, 2 letters, 3 letters or n letters of every word - these are called n-grams. For example, if the token is ‘Smith’, then any record which contains ‘Smith’ after tokenization will be grouped under this token.  Ofcourse, we need to keep a threshold on the size of the blocks because ‘Smith’ is a very common name and so, the block size increases and the pairwise comparisons become expensive and defeats the whole objective of creating blocks.

**How did we build our blocks ?**

1. The fields we were dealing with included first name, last name, company name,  work number, mobile number, phone number and email.
2. We built tokens on the whole of first names and last names after removing the salutations, punctuation and lower-casing them. For example, Mr. John  and john would have the same token - john
3. We also created tokens on the first 5 letters of the names. This is to make sure that two names having the same starting patterns would be captured under the same block. For example, freshworks and freshwrk would be in the same block.
4. The last six digits of the work numbers and mobile numbers. This is to make sure that state/country codes at the beginning and the patterns of the phone numbers does not affect block formation. For example - +91 (697)-232-1232 and 6972321232 would be in the same block.
5. For company names, we removed the stop words like company, inc, corp, llc, ltd,pvt etc. So Apple Inc. and apple pvt ltd would be under the same block.

1. **Block cutoff -** Suppose, let us say we created block on a token ‘john’, now if john is a common name, there will be lots of leads under the same block, and if this number ranges in the millions, then the whole purpose of creating blocks to decrease the search space is defeated. We created a threshold for the size of the block to be 50 for this very reason. So, if any block has  a size greater than 50, then the block will not be formed based on that token. Mind you, the same leads can be paired together under a different block token which might not be that common.

1. **Smart Block Selection -** Even after implementing a block cutoff, we found that some of the blocks which were larger in size (high number of record ids) had a lower probability of containing record pairs that were duplicates than when the blocks were more unique (smaller in size). But by doing this, we might be missing out on actual duplicate in the larger blocks also. To counter this issue, common token methods on the blocks were used. Finally, to have the best out of both methods, we selected blocks from both methods combined (ensemble technique) to create record pairs and at the same time, obtain record pairs which have higher probabilities to be duplicates.

**Phase 2**

**Model -** We employ a machine learning based algorithm (<https://www.expertsystem.com/machine-learning-definition/>) which learns the non linearity of the patterns and the thresholds of the features to distinguish a pair of records as duplicates or non duplicates. In particular, we used a random forest model (a scikit learn python package - <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>) to capture the above nonlinearity and also to solve the problem of overfitting i.e. the model is not too specific to the patterns in our training data but can distinguish records in a much more generic way. Our model is robust to find duplicates even if there are spelling mistakes, phonetic matches, empty fields, punctuations, salutations, field mismatches, abbreviations, variations on phone numbers (state code, area code etc.). To solve the problem of having tagged data, we use a semi active learning technique where we first tag the data based on the matching criteria on the emails of two records. Once the model is trained on this data, it is used to predict on a different test set. This is manually curated and fused with the training data for retraining the model. The aim is to feed those patterns for both duplicates and non duplicates into the model which might have been missed out using the heuristic.

1. Building training and validation dataset (<https://machinelearningmastery.com/difference-test-validation-datasets/>)
   1. Lack of a golden dataset - Initially an unsupervised problem, we made it supervised by introducing a tagging heuristic.
   2. The tagging rule - if emails of two records match, tag them as a duplicate pair, else as a non duplicate pair.
   3. Problems with such kind of a heuristic - pairs with different emails might be very different while pairs with same emails might be very similar.
   4. The differentiation learnt by the model is a very easy one with a visible distinction. The predictions for cases where the records are only slightly different and still are non duplicates will be not as expected. Hence the tagging is done on pairs generated within the blocks. This will solve the issue to a certain extent.
2. Type of Model used - Random Forest (<https://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/>) - because of different data types, captures non linear patterns better than , lets say, a logistic model, less prone to overfitting.
3. Recurring training phase (<https://en.wikipedia.org/wiki/Active_learning>)
   1. We train the model with the current tagging mechanism, test it on a different set of data, identify those pairs where the predictions are going haywire because of the tagging, correct the tags, pass these records into the training dataset and retrain the model.
   2. When we tag the data based on a heuristic, there are cases where the pairs are duplicates but the tags are the opposite. So, when the model is trained on this kind of a dataset, the weights it learns on the different features and the patterns it picks up might not be the correct ones and hence, the predictions might go wrong for some test dataset. So in active learning , we pass a bunch of test data to a human who looks at the pairs manually and tags them. This removes the dependency on the heuristic. When this human curated data is fed into the model and retrained, the weights learnt on the features are much more reliable. We can also then confidently say the tags are correct.
4. How do we validate the model -
5. AUC - measures the predictive power of the model. ([Understanding AUC - ROC Curve – Towards Data Sciencehttps://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5](https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5))
6. Average rank - in the test dataset, for a given duplicate pair, we see if, for one of the records, whether the model places the other record as high as possible in terms of the prediction scores calculated on all the other records.
7. Average False Positive Rate ((<https://en.wikipedia.org/wiki/False_positive_rate>)) - in the test dataset, for a given non duplicate pair, the prediction scores should be low (based on a threshold)
8. Manual validation - We maintain a list of cases of duplicates and non duplicates which we expect model to identify. This list is curated based on business needs.

**Phase 3**

1. **Storing in graph** - Vertex represents a record in the database which has one more duplicates. An edge between two vertices stores the score predicted by the model, which represents how strong a duplicate pair they are.

1. **Showing duplicates to user** - To fetch the duplicates of a particular record, graph is queried using that record id to get the connected vertices. Because vertices are uniquely identifiable, duplicates of a particular record is fetched in constant time. The edge id encodes the ids of the two vertices it is connecting. This helps in fetching the duplicate record from the edge id itself without actually retrieving the connected vertex. While showing duplicates to user, strong duplicates are shown on top. So, duplicates are sorted based on model score before displaying. Since the score is stored in edge itself, it helps in fetching the strongest duplicates at the time of graph querying itself.

* **User Feedback -  The user has choice to tag some suggestions as non duplicates. There is also a mechanism which takes in those pairs which are merged by the users as duplicates. This feedback dataset is used to retrain the model. This is essentially called as active learning.**