RNI Evaluation & Configuration Guide

Version 0.3

About Basis Technology

Verifying identity, understanding customers, anticipating world events, uncovering crime. For over twenty years, Basis Technology has provided the underlying analytical components enabling businesses and governments to tackle some of their toughest problems. Our Rosette™ text analytics platform employs a hybrid of classical machine learning and deep neural nets to extract meaningful information from unstructured data. Autopsy, our digital forensics platform, and Cyber Triage, our incident response tool, serve the needs of law enforcement, national security, and legal technologists with over 5,000 downloads every week. KonaSearch delivers natural language search of every field, object, and file in Salesforce all from the same index.

Company headquarters are in Somerville, Massachusetts, with branch offices in Washington, London, Tel Aviv, and Tokyo. For more information, visit www.basistech.com.

Basis Technology Corp.
1060 Broadway
Somerville, MA 02140
T 617.386.2000
F 617.386.2020
E support@rosette.com
http://support.rosette.com

Table of Contents

1. Introduction

As the rate of data creation increases each year, the importance of effective search tools increases. Names continue to be one of the key identifiers for individuals and organizations. Especially when record identifiers (like national ID numbers or customer IDs) are not available, names are often thought of as the next best data signal to match on. Names are vitally important data points in financial compliance, anti-fraud, government intelligence, law enforcement, and identity verification. Yet matching names can be challenging when your data includes anomalies such as misspellings, aliases or nicknames, initials, and non-Latin scripts -- which it invariably will.

Rosette addresses these issues with a linguistic, A.I.-based system that compares and matches the names of people, places, organizations, as well as addresses, dates and other identifiers despite the many variations each of these are likely to have. Built by linguistics experts, our match technology is unrivaled in its ability to connect entities with high adaptability, precision, and scalability. With fluency across 18 languages and a deep understanding of the linguistic complexities of names, Rosette is the first choice for record matching when names are a key signal.

1.1. Who is this guide for?

Name matching is a fairly simple concept to understand, but it can be challenging to evaluate and optimize for specific needs. As leaders in name matching, with experience working side by side with customers, we have created this guide to help Rosette adopters maximize its potential and quickly deliver value. The purpose of this document is to provide a deeper and practical understanding of RNI, present best practices for conducting name evaluations and testing, and educate users on how to tune RNI to improve accuracy and performance.

1.2. How to use this guide

This guide assumes you are familiar with the basics of Rosette Name Indexer (RNI) as this guide is mostly focused on using the RNI Elasticsearch plugin for evaluation and configuration. You should be comfortable indexing and querying records with Elasticsearch or making calls to the Rosette name similarity API. One goal of this document is to provide a blueprint for conducting effective name evaluations to implement the optimal configuration to support your business requirements. This guide in its entirety acts as a playbook to accomplish this task while providing deeper explanations into the inner workings of RNI. By the end of this guide you should be able to execute the task and methodologies discussed in this document with your own team of engineers and data scientists.

2. Understanding RNI

In this section we will dive under the hood of RNI to understand the overall architecture and explain why and how RNI generates its name similarity score. It is critical to understand how a RNI score is computed in order to identify which algorithms were triggered. This insight will help you understand the configuration settings and how to set them.

2.1. The fundamental matching function

At the most fundamental level, RNI determines how similar two entities are. These entities can be made up of one or more fields that contain Person, Organization, Address, Location, or Date fields. Identifying fields are further broken down into subcomponents called tokens. For example, the tokens in a name might be the

first name, middle name, and last name. Rosette blends machine learning with traditional name matching techniques such as name lists, common key, and rules to determine the best possible alignment and match score between sets of tokens. These granular scores are then combined into an overall record match score. This score can be used to maximize precision or recall depending upon the application.

2.2. Pairwise & index matching

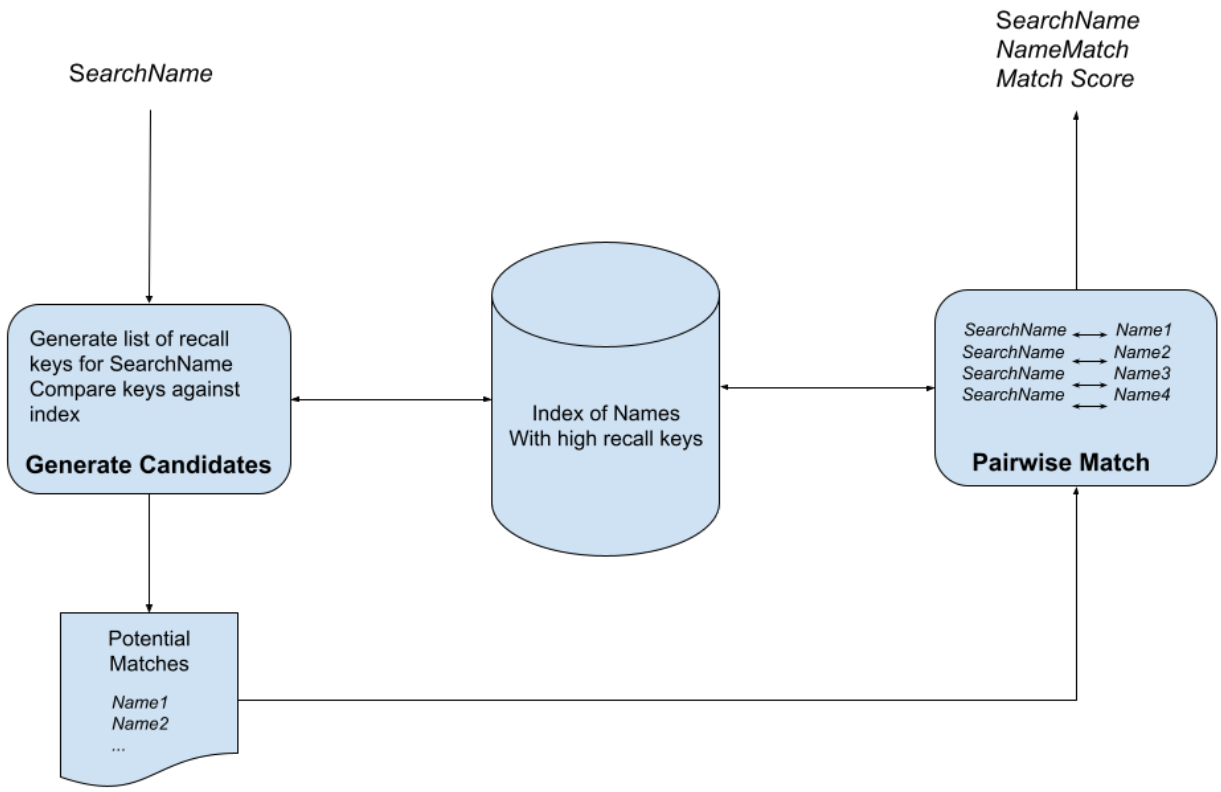
There are two common usage patterns in record matching: pairwise and index.

Pairwise: In pairwise matching, you have 2 records (or even just 2 names) that you are comparing directly to one another. This comparison results in a single “similarity score” that reflects how similar the records are. There are many use cases where pairwise matching is all that is needed. For example, you already have a common key between 2 records and you simply want to check to see if the name fields are acceptably similar. This is common in identity verification applications.

Index: With index matching, you have a single record that you are comparing to a list of records. This usage pattern can be thought of as a search problem. You have a record (or maybe just the name of an entity) and want to search a large list of records to find a ‘match’. As in typical text search applications, the result of the search is a list of possible matches sorted by relevance, or in the case of fuzzy matching, record similarity. This is the case primarily discussed in this document.

2.3. Architecture

Your business sets the priorities for optimizing search to find the right balance between accuracy and speed. You also determine which is more critical to your business: reducing potential matches at the expense of returning false matches or reducing false matches, with the increased risk of missing a potential match. RNI’s patented two-step approach provides the best approach for users to make these critical decisions.



RNI Matching Against an Index

Overview of Name Matching with RNI

1. **Create an index** containing the names you will be searching against. High recall keys are generated from the original name and are stored as the data/records are indexed.

2. **Search** against the index - 2 pass approach

    a. **Generate Candidates**: The first pass is designed to quickly generate a set of candidates for the second pass to consider.

    b. **Pairwise Match**: The second pass compares every value returned by the first pass against the value in the query and computes a similarity score. Multiple scorers are applied in the second pass, to generate the best possible score.

First Pass - Generate Candidates

The first pass takes the name you want to search with and again generates a list of recall keys. These keys are compared against the indexed keys. This approach casts a wide net that reduces false positives regardless of variation (misspelling, language, etc) of name. The maximum number of candidates to send to the second pass is referred to as the **windowSize.** This step creates the list of records to be scored by RNI.

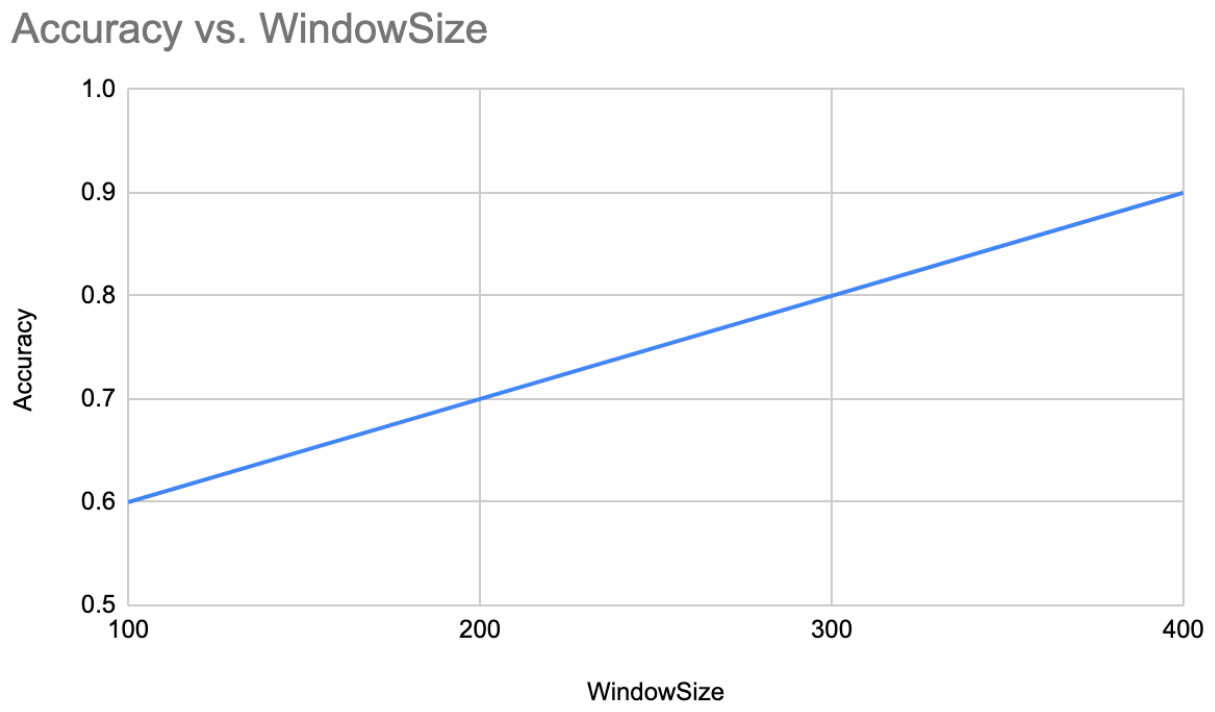
Second Pass - Pairwise Match

The second pass acts as the precision filter which takes the original search record and compares it to each of the recall candidates from the first step. It calculates a similarity score for each name pair using RNI's algorithms and rules. The scores are then sorted into descending order and returned.

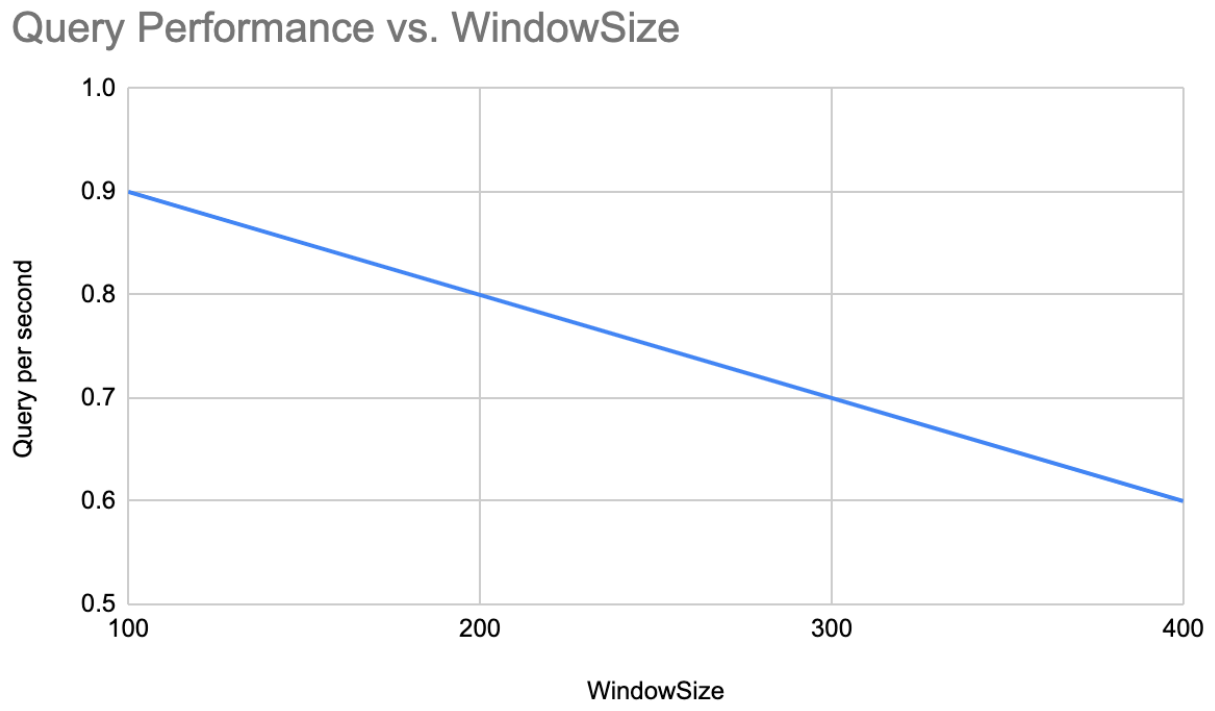
The first pass gives the system the speed necessary for high-transaction environments, eliminating values in the index from consideration. The slower second pass re-compares each selected value directly in their original script, using enhanced scoring algorithms.

2.4. The relationship between accuracy and speed

Within RNI there is a relationship between accuracy and speed. The **windowSize** set on the first step of the search query will largely determine not just how accurate the search is, but also how fast it is. If you set your window size too small, you might miss records you intend to match on, increasing your false negatives. Increasing the window size will likely reduce false negatives, as represented in the following chart.



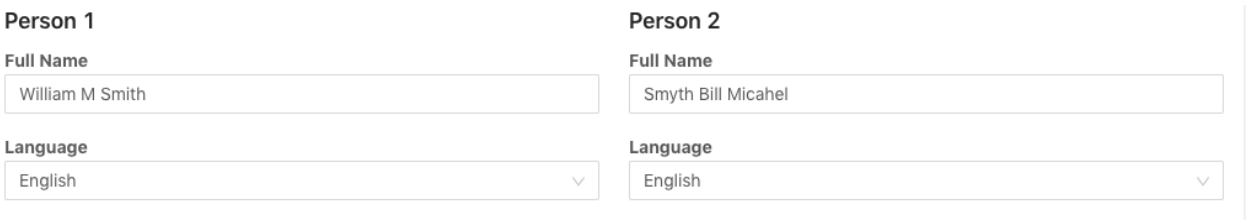
But if you set your window size too large, you will see an impact on performance. A smaller window size results in a smaller list of candidates to consider, improving performance. A larger window size results in a larger candidate pool that requires computation, decreasing performance, as represented in the following chart.



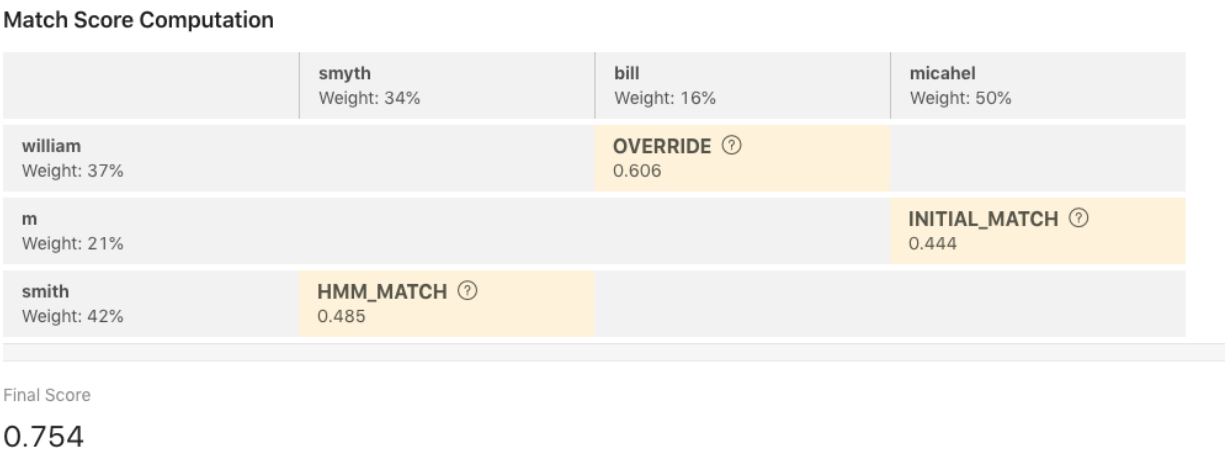
Since **windowSize** is tied to both accuracy and performance, determining the best value for **windowSize** is a critical component when performing an evaluation.

2.5. Understanding match scores

Newcomers to RNI are often confused why some name pairs generate the result they do. Understanding what goes into an RNI score and how to decode it is a key step in executing an effective evaluation. Rosette Match Studio (RMS) is an interactive tool for evaluating and optimizing name matching with RNI by providing insight into the algorithms and parameters used. RMS provides an easy-to-use UI and displays the details of how an RNI score was calculated. Let's walk through an example of an RNI score using the following:



In this example we compare ‘William M Smith’ to ‘Smyth Bill Michael’. The RNI result of these two names is shown below.



Here RMS shows how we arrived at the score of 0.754. There are several things working together to arrive at this score. Let's review each step of the process.

- **Tokenization, normalization, and transliteration:** Before any matching algorithms can be run, each name is transformed into individual tokens that can be compared. This step includes removing stop words, such as 'Mr'., 'Senator', or 'General', and transliterating names into English. This step is more complicated when you are using languages that do not use white space. Your first step should be to determine if tokenization is being done correctly. In this case it is. Also, look to see if the correct stop words have been removed.

- **Token alignment:** Next, every token in Name 1 is matched and scored against every token in Name 2 to find the matching pairs that produce the highest total score. All candidate token pairs are scored to determine the best match alignment. Token scorers are modular, allowing a different scorer to be chosen for each pair. RNI arranges the tokens from the names into a grid. The aligned tokens are shown in yellow. This is referred to as the *token alignment matrix*.

- **Weighted score:** In addition to the token match score, the score displayed takes into consideration the order the tokens are in, the uniqueness of the given token, and the algorithm triggered within RNI. In each yellow cell of the grid you see the rule fired in bold print. In the case of tokens 'smith' and 'smyth' the rule fired is 'HMM_MATCH' that generated a score of 0.485. Underneath the tokens themselves you will see how much weight that token has in the scoring process. This grid explains what name tokens were used in the scoring, what was used to score them and the individual scores of the tokens.

- **Final scoring and bias adjustment:** At the end, all selected tokens and token-pairs are considered together and some final adjustments are made to the score. For example, a penalty is applied if there is a gender mismatch between tokens. Adjustments are often language dependent. The scores are combined to form a final score and are adjusted by a final bias. Final bias is a value added to the final score to help names in different languages result in a similar score.

- **Additional information:** RMS provides more detailed match information by checking the **Explanation** checkbox below the token alignment matrix. This data is also provided in the response of a match query. This information contains low level details of the various tokens and how they were assigned. It will also contain key information regarding the language, script, and language of origin of a name. When dealing with multilingual names this is another area that is critical to understand. Getting the language and script correct is vital to the correctness of a score.

Understanding the token alignment matrix is the key to understanding the RNI similarity score. It allows you to effectively conduct an error analysis for a given name pair. It is important to observe and record how each

token was scored. This information can be thought of as error classes and will be important later when you configure the control parameters.

2.6. Configuration parameters

Individual name tokens are scored by a number of algorithms or rules. These algorithms and rules can be manipulated by setting configuration parameters, changing the final RNI similarity score. There are 100 plus RNI configuration parameters. The *RNI Application Developer's Guide* describes these parameters in greater detail and how to modify these parameters. In RMS, you can adjust a subset of the most commonly modified parameters, seeing how they change the match scores and optimizing for your specific use cases. In Parameter configuration [12], we discuss techniques for setting parameters to configure your RNI installation.

3. The case for evaluation

Consider the following questions:

1. How well is our name matching working?
2. Should I upgrade?
3. When I upgrade, how does it compare to my current version?

These questions and many others can only be answered by performing a proper name evaluation. An evaluation arms you with quantitative data you need to inform key stakeholders on the accuracy and performance of your name matching tool. Additionally, it measures how your name matching is behaving as new data sources enter your system and provides guidance on how to adapt the tool to changes.

3.1. The unique problems with name matching

Matching names is a challenging problem because they can differ in so many ways, from simple misspellings, to nicknames, truncations, variable spaces (Mary Ellen, Maryellen), spelling variations, and names written in different languages. Nicknames have a strong cultural component. The terminology itself is problematic because matching implies two things that are the same or equal, but name matching is more about how *similar* two entities are. Once you have a measure of similarity, you may need additional rules or human analysis to determine if it is a *match*. It is important to understand these challenges and to address them in your evaluation.

3.2. The importance of quantifiable, repeatable evaluation in your environment, on your data

Implementing a robust, formal methodology is the ideal way to reach accuracy and performance goals that satisfy stakeholders reliably. It's important that the evaluation is performed on your data and in your environment. Generic test data may not have the same name variations and language distributions as your data. The only way to reliably test and improve performance is by using test data that is the same or similar enough to your production data. A formal approach provides consistency across evaluations and allows for small iterative exercises to be measured and to capture information to help improve the methodology.

3.3. Relationship between configuration & evaluation

Out of the box RNI comes equipped with its algorithms optimized and parameters set based on generalized name testing. While this is a great starting point, each customer's data and needs are different. One

customer might value recall, another precision. As a result the default RNI settings might not be ideal for your environment. A primary goal of running an effective evaluation should be to identify an optimal configuration based on your data and business requirements.

4. Planning an evaluation

4.1. Setting yourself up for success

Effective name evaluation should be treated with the same attention and effort as any other investment. Many efforts have failed or have been delayed simply because of a lack of appreciation for this size and importance of this task. You have to have the right team and a clear understanding of your business requirements.

Building the right team with the right skills is paramount. It should consist of engineers for building automation tasks, data scientists/business analysts to review data and analysis results, and project managers to track progress, generate reports and manage team members.

At the start of the evaluation, identify your business goals and identify key questions you need to answer. These goals will be used to shape your evaluation and also when defending the results of your analysis and stakeholders.

4.2. Evaluation data

TBs or even GBs of information are not required to effectively test accuracy and performance. Experience has shown us that for some tasks thousands of records of data provides the same information as millions of records. While it may seem advantageous to use as much data as possible, we believe a more pragmatic approach is often best. Basis advises organizations to curate a data set that is representative of the data they will experience in an operational environment. A key distinction needs to be made between data for testing accuracy and data for testing performance.

Data for evaluating accuracy:

Data used to measure accuracy should include a wide variety of phenomena that make name matching challenging, including misspellings, aliases or nicknames, initials, and non-Latin scripts. Applying organizational domain knowledge to curating name data that contains specific phenomena found in your real world cases is an ideal starting point for crafting this data set.

Your data for testing accuracy should contain labeled or annotated data. This is often called *gold data*, referring to the accuracy of the training set's classification for supervised learning techniques. For name matching, it is a list of name pairs that are considered a match. You can't calculate accuracy without labeled data. Since assigning classification labels to data can be subjective, you should use multiple annotators on the same data set, determining positive and negative name matches. Establishing a set of annotation guidelines for scoring a classification is necessary as it provides consistency when classifying the data.

JRC-Names is an industry standard, open-source gold data set. It is a named entity resource, consisting of large lists of names and their many spelling variants. Variations include misspellings, extra or missing fields, concatenations, reordering, extra syllables. This data set is an excellent representation of the challenges found in real world applications. You still have to determine if this data set reflects your business data but is a great place to start.

Data for evaluating performance:

Performance data does not need any classification or gold labeling. Since you will be examining query execution time, you need only hundreds to thousands of records. The key with performance data is to have an index size representative of your production index.

4.3. Measuring accuracy & performance

4.3.1. True positives, False positives, and False negatives

A major benefit of RNI is that one can define a threshold for name matching, thus emphasizing what is most relevant to the use case. For index name matching, consider the case where a query name is expected to match one and only one name in the index. If there are three names returned above the threshold, including the correct match, then one name is a true positive (TP), two names are false positives (FP), and there are no false negatives (FN). If the correct match is not returned above the threshold, the number of false negatives will be one.

4.3.2. Recall and precision:

Recall is an indication of how many hits were correctly returned. It is defined by the following formula:

$$R = TP/(TP + FN) \tag{1}$$

In the above example where the correct match is returned along with two other matches, the recall is 1.0 since there are no false negatives.

Precision is an indication of how many relevant hits are found in the returned matches. It is defined as:

$$P = TP/(TP + FP) \tag{2}$$

In the above example where the correct match is returned along with two other matches, the precision is $1/(1 + 2) = ⅓$.

4.3.3. Accuracy

F1 score is the industry standard for calculating accuracy as it aims to measure a test's accuracy. It is the harmonic mean of precision and recall and is calculated by the following formula.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3}$$

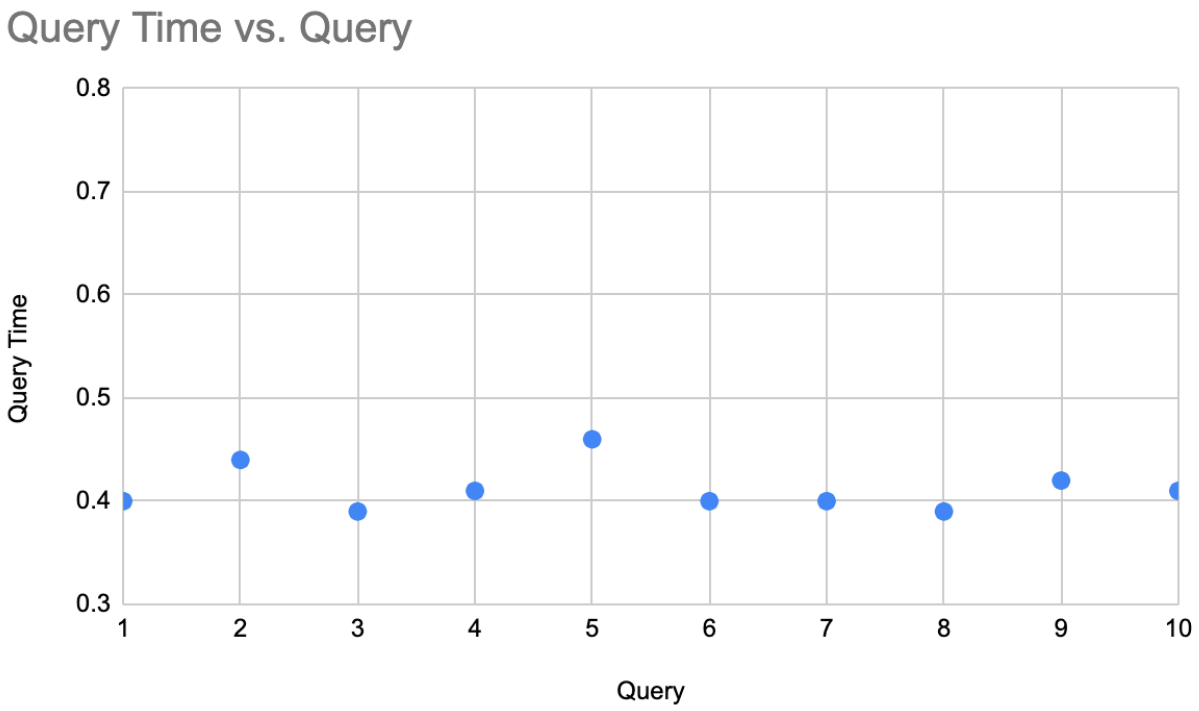
Given the recall and precision calculations mentioned above, you would have an F1 of ½. Armed with this information you can now evaluate multiple configurations or versions of RNI to determine how well it is performing and identify the best possible configuration.

4.3.4. Performance

By far the most popular metric to capture for a performance evaluation is 'seconds per query'. This will tell us how long a query takes and addresses requirements where specific search response times are needed. Many factors go into query performance where query complexity, index size, Elastic configuration are the main ones. How Elastic configuration impacts performance will be discussed in a later section.

CPU utilization is another statistic you will likely want to measure. Depending on your use case, parallelization may be used to increase the throughput of the system.

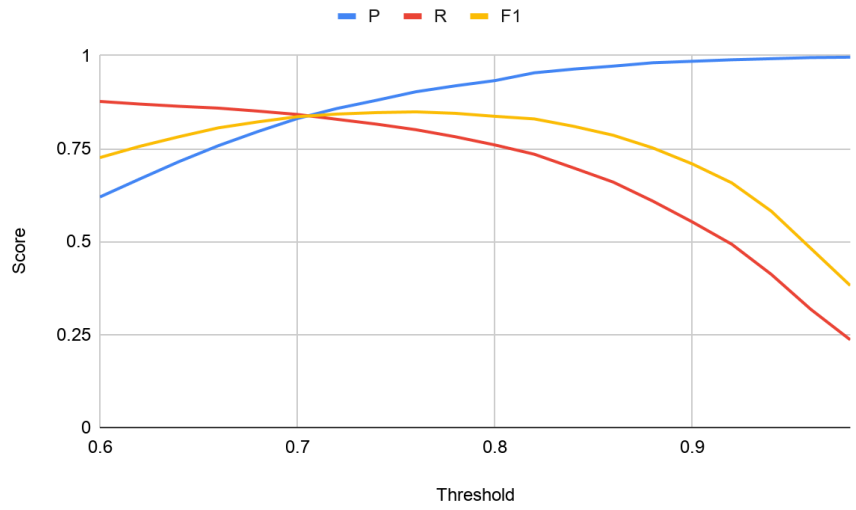
When it comes to reporting performance it is advisable to report both averages as well as actuals in a scatter plot or something similar (see chart below). This will inform the behavior of performance over time for cases where bulk processing is done.



5. Establishing your match threshold

A key feature of RNI is that it returns a normalized score between 0 and 1 indicating how similar two names are. This makes integrating RNI into existing workflows straight forward. However, it leads to the inevitable question: "*What threshold should I use*?" Without quantitative error analysis, this question is difficult to answer. Now that you have learned how to measure the accuracy of your evaluations, you can leverage additional automation to calculate an optimal threshold. By adding different threshold values to your automated evaluation tool, you can process your evaluation data against the various thresholds, calculating precision, recall, and F1 for each. The end result of this process is clear understanding of an optimal threshold.

As one can see from the following graph, the threshold can be adjusted to favor precision or recall. With a high threshold, false positives will be less likely, leading to a higher precision. With a low threshold, false negatives will be less likely, leading to a higher recall. For applications involving border checking you may be inclined to favor recall to avoid allowing potential threats into a country. Conversely, for Know Your Customer(KYC) applications, leaning your threshold towards precision may be best to reduce false positives. Your choice of threshold should ultimately be based on an analysis of accuracy as a function of threshold for your particular data, as well as your business requirements.



6. Configuring for Accuracy

6.1. Setting entity type and language/script

Where possible, you should always set the entity type and language arguments. While these arguments are optional and RNI is flexible enough to work without them, you will see better results when you use them.

RNI has different algorithms for matching PERSON and ORGANIZATION names. If you don't specify the entity type, RNI will use a simplified rule set of the PERSON entity type. This may lead to scores that do not meet expectations.

Similarly, when language and script are not provided RNI will attempt to guess what the language is. Given the similarity across some languages like Ukrainian and Russian or Japanese and Chinese, the guess may be incorrect. The language determines what model and rules are applied in the scoring and if the language/script are wrong it will likely lead to less accurate RNI scores.

It is strongly recommended that the entity type and language/script are determined and set during indexing and querying processes.

6.2. Token Frequency

Name tokenization is the process of breaking up a name into pieces such as first name and last name. Use the list of names you have indexed and any historical queries you have saved to determine how often a token appears in your name universe. RNI processes full names, but the underlying algorithms work on a token level. Knowing how often a name token appears can provide insight into how to improve accuracy. For English names tokenization is as simple as finding the white space in a name. For languages that don't use whitespace, you may need to incorporate Rosette Base Linguistics (RBL) which provides tokenization for 33 languages. The end result of this process should look like the following table.

TOKEN	FREQUENCY
Smith	76
INC	54

TOKEN	FREQUENCY
of	32
John	28

Token frequency lets us accomplish a few things.

1. It identifies potential candidates for stopwords. These are words you don't want to be used in the comparison. In the above example "INC" and "of" are frequently found. These words should have no real impact on the score as they are common and generic. RNI includes lists of stop words which can be modified, leading to a higher quality score.

2. RNI takes into consideration the uniqueness of a name when it calculates the final score. Using this output you can retrain the model on customer-specific data to help improve results.

3. Finally, this analysis helps identify data integrity issues. RNI is designed to work on name/entity data. Data migration and ingestion are complicated processes and often don't work perfectly. It may be the case that web URLs or some other unintended transformations may have found their way into your data set. This will help identify potential issues and define actions to clean up any inconsistencies.

6.3. Parameter configuration

Evaluating your data set will provide insight into the scoring for each name pair and can result in a breakdown of errors for each name pair. This information can show you which parameters have the largest impact in name scoring with your data and direct you towards the parameters to target for adjustment. Given the large number of configurable match parameters in RNI, it is advisable to start with just a few. Once again, this process can be done via automation, as was done with determining the optimal threshold. Take a range of values for a small set of parameters and run your evaluation against all possible parameter configurations to determine the accuracy for each configuration. The end result of this process will be a threshold value along with the optimal parameter configuration. Using the data processing techniques from the previous section, you can extend it to include finding how often some of the following phenomena occur in your data sets.

6.3.1. Commonly Modified Name Parameters

Given the large number of configurable name match parameters in RNI, you should start by looking at the impact of modifying a small number of parameters. The complete definition of all available parameters is found in the `parameter_defs.yaml` file.

Parameter	Description	Behavior
`conflictScore`	The score that is assigned to unmatched conflict tokens	Increasing leads to higher final score
`initialsConflictScore`	The score that is assigned to unmatched conflict initials	Increasing leads to higher final score
`initialsScore`	The score that is assigned to an initial matching a token	Increasing leads to higher final score
`initialismScore`	Score assigned to initialism matching a name	The score that is assigned to an initial matching a token
`stuckInitialScore`	Score applied when initial is "stuck" to previous token	Increasing leads to higher final score
`deletionScore`	Score applied to an unmatched token when surrounding tokens are matched	Increasing leads to higher final score
`outOfOrderDeletionScore`	Score applied to an unmatched token when surrounding tokens are also unmatched	Increasing leads to higher final score

Parameter	Description	Behavior
`reorderPenalty`	Penalty applied to matching tokens with different positions	Increasing leads to lower final score
`initialsDeletionPenalty`	Multiplier on token deletion score when deleted token is an initial	Increasing leads to higher final score
`genderConflictPenalty`	Penalty applied when name genders don't match	Increasing leads to lower final score
`crossLanguageGenderConflictPenalty`	Penalty applied when name genders of different languages don't match	Increasing leads to lower final score
`boostWeightAtRightEnd`	Boost applied to tokens at the right end of the name (i.e. surnames in English)	
`boostWeightAtBothEnds`	Boost applied to tokens at either end of the name (i.e. less weight for middle names in English)	
`adjustOneSideDeletionScores`	Multiplier on the token deletion score when all deleted tokens are on one side of the name	Increasing leads to higher score
`reorderCorrection`	Boost to final score if one name's tokens are a reordering of the others	Increasing leads to higher final score
`finalBias`	Helps normalize scores	Increasing leads to a higher score for ALL names

The following examples describe the impact of parameter changes in more detail.

Token Conflict Score `conflictScore`

Let's look at the two names: 'John Mike Smith' and 'John Joe Smith'. 'John' from the first and second name will be matched as well the token 'Smith' from each name. This leaves unmatched tokens 'Mike' and 'Joe'. These two tokens are in direct conflict with each other and users can determine how it is scored. A value closer to 1.0 will treat 'Mike' and 'Joe' as equal. A value closer to 0.0 will have the opposite effect. This parameter is important when you decide names that have tokens that are dissimilar should have lower final scores. Or you may decide that if two of the tokens are the same, the third token (middle name?) is not as important.

Initials Score (`initialsScore`)

Consider the following two names: 'John Mike Smith' and 'John M Smith'. 'Mike' and 'M' trigger an initial match. You can control how this gets scored. A value closer to 1.0 will treat 'Mike' and 'M' as equal and increase the overall match score. A value closer to 0.0 will have the opposite effect. This parameter is important when you know there is a lot of initialism in your data sets.

Token Deletion Score (`deletionScore`)

Consider the following two names: 'John Mike Smith' and 'John Smith'. The name token 'Mike' is left unpaired with a token from the second name. In this example a value closer to 1.0 will not penalize the missing token. A value closer to 0.0 will have the opposite effect. This parameter is important when you have a lot of variation of token length in your name set.

Token Reorder Penalty (`reorderPenalty`)

This parameter is applied when tokens match but are in different positions in the two names. Consider the following two names: 'John Mike Smith', and 'John Smith Mike'. This parameter will control the extent to which the token ordering ('Mike Smith' vs. 'Smith Mike') decreases the final match score. A value closer to 1.0 will penalize the final score, driving it lower. A value closer to 0.0 will not penalize the order. This parameter is important when the order of tokens in the name is known. If you know that all your name data stores last name in the last token position, you may want to penalize token reordering more by increasing the

penalty. If your data is not well-structured, with some last names first but not all, you may want to lower the penalty.

Right End Boost/Both Ends Boost (`boostWeightAtRightEnd`,`boostWeightAtBothEndsboost`)

These parameters boost the weights of tokens in the first and/or last position of a name. These parameters are useful when dealing with English names, and you are confident of the placement of the surname. Consider the following two names: "John Mike Smith' and 'John Jay M Smith'. By boosting both ends you effectively give more weight to the 'John' and 'Smith' tokens. This parameter is important when you have several tokens in a name and are confident that the first and last token are the more important tokens.

6.4. Token Weighting

Users have the ability to classify specific tokens as 'low weight'. These are tokens that you don't want removed from the scoring of the name but should not carry the same weight as other name tokens. Given the following organization name example: 'XYZ Systems'. The name token 'Systems' is commonly used in company names. This token will help contribute to matching against other organizations names likely creating false positives hits. Users can identify 'Systems' as a low weight token and add it to `lowWeighttokens.datafiles` and RNI will decrease its importance when scoring names that contain 'Systems'. In the above case the result will put more emphasis on the 'XYZ' name token.

6.5. Incorporating and optimizing field weights

Adding additional search signals to your query greatly impacts the accuracy of your results. RNI can also incorporate search signals of field types that it does not support. Fields like 'gender' or 'occupation' are not supported by RNI but can be used in the rescoring process to help optimize accuracy. While assessing your data a key step would be to evaluate the various fields and determine how sparse your data is. If many fields are optional when collecting data, this can result in missing or null data. Knowing how sparse your data is will give insight on the best way to craft your rescoring process. Users can control how missing or null data is handled by the rescoring process. By default, if a queried-for field is null in the index, the field is removed from the score calculation, and the weights of the other fields are redistributed. However, you can override this behavior by using the **score_if_null** option to specify what score should be returned for this field if it is null in the index document.

The next step is to determine how fields should be weighted by the rescoring algorithm. Each field can be given a weight to reflect its importance in the overall matching logic. Determining the ideal weight can be a challenge that requires understanding your data. How much do you trust the quality of each field and what is its importance to your business goals? It is advisable to think about a range of weights for each field, giving primary fields like person name and organization name high ranges and fields of lesser importance and quality lower ranges. Let's look at an example to illustrate the impact of field weighting.

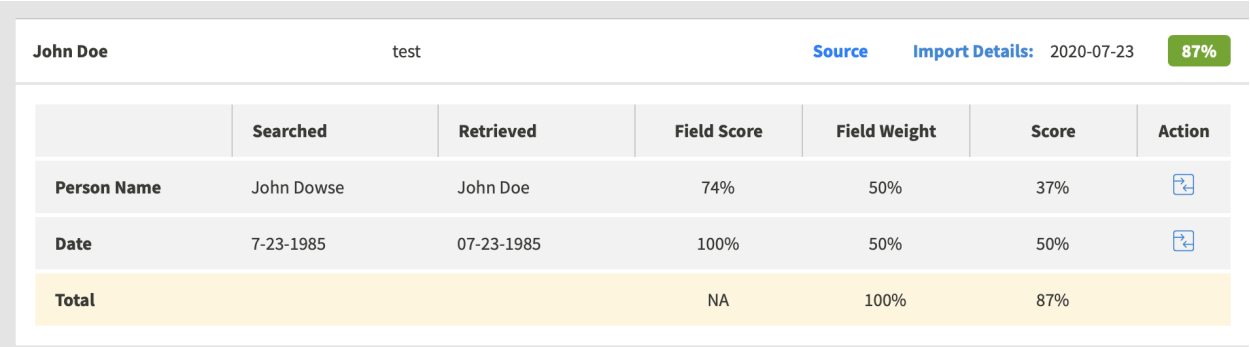
Given the following two records:

RECORD 1 RECORD 2

NAME: John Doe NAME: John Dowse

DOB: 07-23-1985 DOB: 07-23-1985

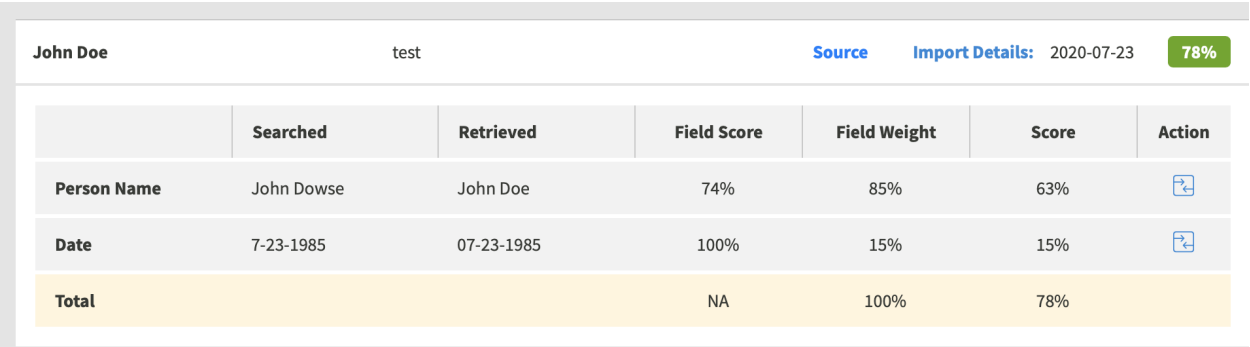
If the 'NAME' and 'DOB' fields are given equal weighting, the score in this case is 87% between the two records.



John Doe test Source Import Details: 2020-07-23 87%

	Searched	Retrieved	Field Score	Field Weight	Score	Action
Person Name	John Dowse	John Doe	74%	50%	37%	
Date	7-23-1985	07-23-1985	100%	50%	50%	
Total			NA	100%	87%	

This is a strong RNI score but likely a false positive as the last names are noticeably different. Now let's give the 'NAME' a weight of 85% and 'DOB' 15%.



John Doe test Source Import Details: 2020-07-23 78%

	Searched	Retrieved	Field Score	Field Weight	Score	Action
Person Name	John Dowse	John Doe	74%	85%	63%	
Date	7-23-1985	07-23-1985	100%	15%	15%	
Total			NA	100%	78%	

Let's assume we apply a threshold of 80%. With this weighting configuration our match score falls below the threshold, turning the once false positive into a true negative, thus improving the accuracy.

Identify the ranges for your field weights, then update your evaluation framework to include various configurations when identifying an optimal accuracy threshold.

7. Configuring for maximum performance

7.1. Pairwise

Pairwise matching in RNI has the benefit of being completely stateless, making it possible to optimize performance through parallelization. You are limited only by the concurrency limit on Rosette Server/Cloud or your client application.

7.2. Index

First you must define what you are trying to optimize. Latency? Throughput? In RNI latency is often thought of as "seconds per query" or how long a query takes to execute. Throughput is usually defined as "how long does this list of queries take to run?" Inside RNI, latency and throughput depend on many factors:

- Size of the index and configuration of the Elastic cluster
- How the query is constructed
- windowSize

- Query methods implemented

We will examine each of these below.

7.2.1. Index size and Elastic configuration

How many nodes do I need? This is a common question. It is very important to performance in addition to indicating the resources you need to provision. To answer, start by examining the data you intend to index. First identify the fields that will be in the Elastic document. Then identify the fields which are of the RNI field type. Use the Elastic mapping for your index to complete this process. Once calculated, determine the size of each field, the more precise the better. RNI fields generate 4 times as much metadata on indexing. The metadata helps with the recall of candidates to avoid false negatives. Given the total number of records to be stored in the index, you can now calculate the total size of your data. The table below shows an example.

Calculate index data size:

Size Per Field	Number of Fields	RNI Field	Number of Records	Total Size
0.03	2	YES (x4)	41M	9.9GB
0.03	6	NO	41M	7.4GB
Total ES Size	17.3 GB			

You also need to consider the growth projections of your data. If your data is expected to grow by 600,000 records per year, you will want to calculate those values as well.

Calculate index size growth projections:

	2021	2022	2023	2024	2025
Number of Records	41.6M	42.2M	42.8M	43.4M	44M
Total Size	17.5GB	17.7GB	18GB	18.2GB	18.5GB

A final consideration is backup or resilience within your cluster. Your tolerance defined by your risk and backup policies will determine the number of replicas to consider. Each replica increases the required storage capacity.

$$TotalSize + Resiliency(1 + numberofreplicas) * indexsize \tag{4}$$

Year	1 replica	2 replica	3 replica
2021	35GB	53GB	70GB
2022	35.5GB	53.1GB	71GB
2023	36GB	54GB	72GB
2024	36.4GB	54.6GB	73GB
2025	37GB	55.5GB	74GB

You should aim to keep the average shard size between a few GB and a few tens of GB. A general rule of thumb is to have 1 shard and 1 replica per node. This allows for metadata and models to be in memory at the time of query for optimal performance. Continuing our example you would have the following cluster.

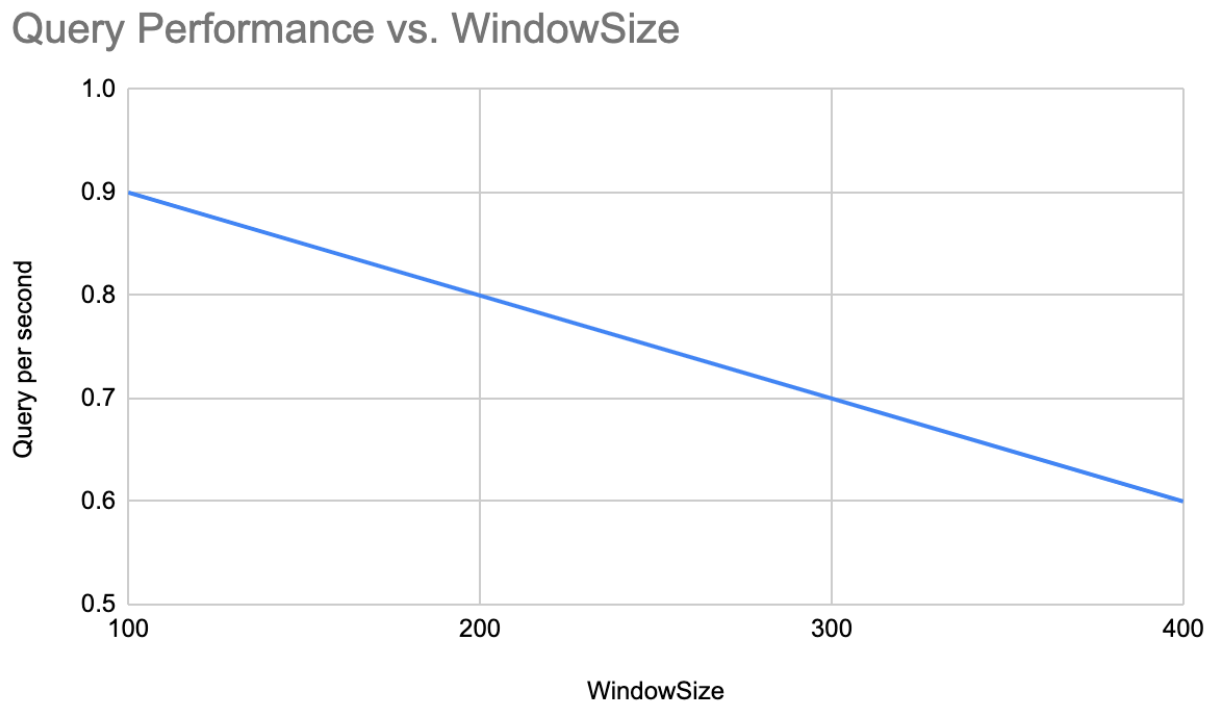
Number of Nodes	Total Shards	Replicas	Size per Shard
3	3	1	15GB
*4	4	1	10GB
5	5	1	8GB

*recommendation

Performance expectations for the recommended configuration should be between 500ms and 1.5s depending on RNI configuration options. Actual performance may vary. The previous recommendation is intended to be a starting point. Performance analysis should be conducted on actual hardware and data to confirm.

7.2.2. WindowSize

The query rescorer executes a second query only on the Top-K results returned by the query and post-filter phases. The number of docs which will be examined on each shard can be controlled by the windowSize parameter, which defaults to 10. As mentioned previously, the windowSize set in the first step of the query has a direct impact on the performance of RNI. Setting the windowSize to a small value limits the number of candidates sent to the second pass. Note that the windowSize will be applied to each shard in your index. The chart below illustrates the relationship between performance and windowSize.



7.2.3. Query construction

The query construction in the first pass plays an important role in overall performance. RNI performance is tied to how many candidates the first pass generates. There is a fundamental difference between a query with a bunch of terms combined using an OR and a query with an AND clause. A query that uses boolean logic containing ORs will fill up the windowSize more quickly than a query that is more selective. The more specific you can make the first pass query, the smaller the number of candidates selected, reducing the time spent in the second pass.

7.2.4. Query methods

There are essentially two ways to query an index:

- Single queries

- Batch queries

As the name suggests, single queries are fired off individually while batch queries execute a set of queries together. A query is sent to Elastic and waits for a response to be returned before continuing. Elasticsearch does offer a bulk API that will process multiple operations into a single call, but some of these functions are single threaded which might not improve your throughput. To improve throughput parallelize the client that creates and sends the query, in accordance with its implementation to achieve greater throughput.

8. Common questions and solutions

This section briefly describes common problems we encounter with our customers and offer a quick explanation and guidance on how to address them.

Question: Why is a name I expect to be in my hit set missing?

Explanation: A recall issue is when an expected name is missing from the result and occurs when the name is not found during the first pass of the query. This means the windowSize is too small or the windowSize filled up, which can happen for very popular names like “Smith”. Try the following steps:

1. Confirm the first pass query finds your name/record by executing just the first pass.

2. Expand the windowSize until you see your name/record

Question: Do I need to index all my data? How do I keep my index up to date?

Explanation: No, you only need to index data that you will be using to match records. In many applications/systems you have a primary data store that stores all information on a given record. You do not need to duplicate that data in Elastic. Focus on identifying key fields that help determine record matching. As your primary data store is updated, you can update those same records in Elastic. You do not need to do a full data update after each change in your data.

Question: I have first name, last name, and middle name in separate fields, do I need to use full name? Why?

Explanation: RNI matching works best on full names. Using partial names will likely increase the number of false positives and lower your accuracy. You can modify the token weights to better represent the impact of each field to get the desired results.

Question: What should I do if I don’t know the entity type?

Explanation: It is highly recommended that entity type and language are set when indexing and querying RNI names. In some cases entity type might not be known. You can leave entity type blank or use the NONE type. This will make matching more generic and will impact your results. For example, gender mismatch will no longer apply because the entity is not a PERSON. As a result you might need to lower your match threshold to avoid increasing false negatives as scores may be lower overall as a result.

Question: Why am I getting an unsupported language error when I submit a query?

Explanation: RNI supports 18 different languages. When indexing or querying names that RNI does not support, an error message will be returned and the record will not be indexed or the query will not return a hit. To better handle these cases you might consider using Rosette’s language identifier to determine the language of the name. If the language is not supported by RNI you can send the index/query request to a separate queue to be handled separately.

Question: I have two similar names like “Jay” and “Jude”, but RNI doesnt match them, why is that?

Explanation: RNI matches phonetics and generally doesn't calculate edit distance. “Jay” and “Jude” will generate different metaphones so they won't match in the first pass, then likely the HMM won't score them so highly because the training pairs won't have had matches that are similar. Two names looking somewhat similar doesn't mean they'll score highly in RNI.

Question: I have my own entity scoring algorithm, can I use RNI to generate all the name variations for me then feed those names into my system?

Explanation: This is a classic search anti pattern. You will never be able to generate a list long enough to get the proper recall taking that approach and it would come at a significant performance cost. You accomplish the same outcome by leveraging RNI’s high recall keys and rescoring algorithm that can be built to include additional entity fields to calculate a similarity score.

Question: I have two names that I think are a pretty strong match, why is the RNI score zero?

Explanation: When a name is found as a result of the initial query (first pass) but the windowSize is set to low, names that fall outside the windowSize and fail to make it to the rescore query(2nd pass) get a score of 0.0.

Question: I can configure RNI to successfully match non obvious organization nick names like “Six flags” and “6 flags” or “Forever 21” and “Forever Twenty One”?

Explanation: Yes you can create organization token overrides or equivalence classes to handle these types of cases.

Question: I have a large organization and we have multiple and different uses of RNI, do I need a separate installation to handle each?

Explanation: RNI supports the ability to have individual named parameter universes for each scenario. This allows customers to configure RNI specially for their needs. During runtime you can specify which parameter universe RNI should use.