Independent Address Identification Search Engine for National Statistical Institute Using ElasticSearch

Raya Berova
Insee
raya.berova@insee.fr

2024-09-30

Introduction

In order for survey interviewers to reach individuals, it is essential to accurately identify and geolocate their addresses. Address data, used in a wide range of statistical processes—from census to surveys—is often difficult to process due to inconsistencies, variations in input, and the volume of records.

In matters of address search, many rely on established services like Google Maps or OpenStreetMap. However, these platforms often pose limitations in terms of data control and reliability. Creating a custom address identification search engine provides complete control over the data, addressing concerns about data source transparency and monthly data updates.

A solution employing ElasticSearch (ES), a powerful software used to create and configure search engines, is here proposed to build an independent process for identifying address data for the National Statistical Institute (NSI). Moreover, ES enables text-based address search and supports the storage of geometric objects, considering the spatial aspect of addresses. This approach optimizes both processing time and accuracy by employing a two-step strategy: an initial strict search for precise address identification, followed by a flexible matching phase for addresses not identified in the first step, which accounts for spelling errors and variations in the input.

1 Methodology

The methodology is based on the implementation of ES to handle large-scale address datasets. Data is organized into JSON **documents**, which represent individual entities. To enable efficient searching, the search engine employs **inverted indices**—a data structure that links terms to the documents where they appear. Its distributed architecture supports the rapid search and analysis of vast data volumes, providing near real-time performance [1].

1.1 Address data indices

Two key indices were developed: one for complete addresses and one for street names.

Table 1: Example address index

idAddress	Number	Suffix	Full Address	idStreet	Postal Code
A1	10		10 Boulevard Royal Crescent	V1	W1B4AA
A2	25		25 Road Kingsway Court	V2	W1B4AA
A3	55	bis	55 bis Street Michael's Mount	V3	SW36PE

Table 2: Example street index

idStreet	Type of Street	Name of Street	Postal Code
V1	Boulevard	Royal Crescent	W1B4AA
V2	Road	Kingsway Court	W1B4AA
V3	Street	Michael's Mount	SW36PE

In reality, other variables are present in the indices, including geolocation variables. However, these are not useful for conducting text-based address search. To enable effective searches, it is crucial that the data in the indices is processed in the same way as the addresses being identified. For this reason, the same filters are applied, called **analyzers**, to both the indices and the inputs to the search engine. These include:
- Lowercasing
- Accents handling
- Punctuation handling
- Use of synonym lists to normalize spellings

1.2 Searching process

To search for an address using this engine, the input must include a complete address string, for example, "10 bd Royal Crescent", along with its postal code string, such as "W1B4AA". Multiple queries will then be executed:

1. **Exact Match Query:** A strict query is performed within the town to match the exact address string. The query returns addresses from the index where the *"full address"* field matches exactly with the address provided. It bypasses tokenization, which is applied by default in ES, to ensure precision. This step allows a quick identification of addresses with perfect spelling after applying the analyzers, resulting in a time-saving process. If the address is not identified at this stage, the process moves on to the next step.

2. **Street-Level Query:** This step involves a more complex query with fuzzy matching to identify the street within the town. A fuzzy matching is an efficient approximate string matching technique [2]. Correcting a string with a fuzziness level of 1 means adding a letter, removing a letter, replacing a letter, or swapping two letters. In order for a street to be returned in the search results, at least one of the following conditions must be met:

- Matching the hole street name with a fuzziness level of 1
- Matching one token of the street type or street name with a fuzziness level of 1
- Matching one n-gram of the street name (n = 3, 4 and 5) Each time a condition is met, the street's **score** increases. Additionally, weights have been applied to these conditions, referred to as **"boosts"**. Here, the first condition carries the most weight. The street with the highest score will be chosen. In addition, an external verification outside of Elasticsearch is performed to validate the selected street. If the address provided contains less than 10% of the trigrams from the reference street, the street is considered not correct.

3. **Number and suffix Query:** If the street is identified, the algorithm searches for addresses within the street (*idStreet*). To be returned, the address numbers in the index must match ex-

actly the number extracted from the address to identify. This extraction is performed using a custom-developed **RegEx**. Additionally, a function has been implemented to check the provided address for suffix, and if any are found, the reference addresses with the same suffix will be boosted.

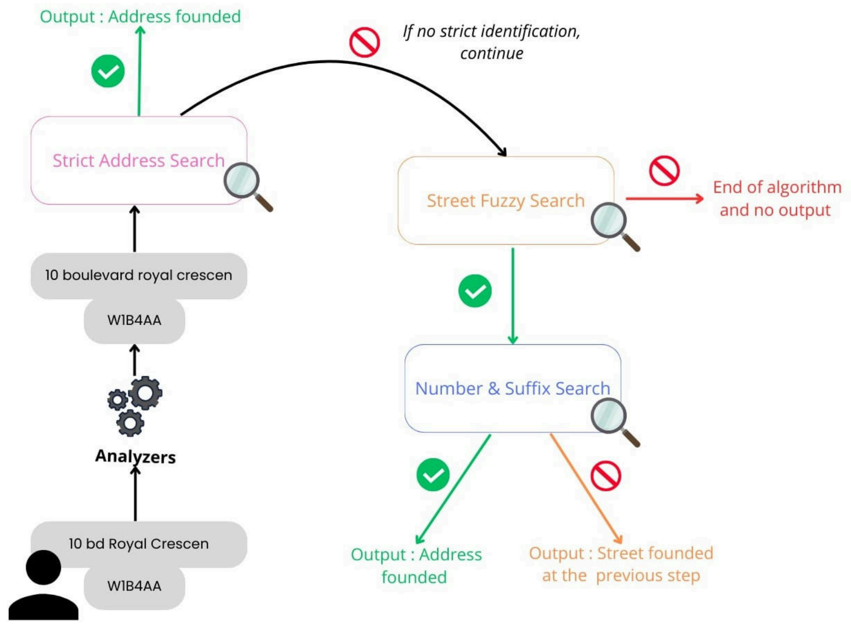

Figure 1. Overview diagram of the full identification process

2 Results and practical application

The ES indices were populated with addresses derived from multiple clean and reliable sources from the NSI. In total, approximately 27 million addresses and 3 million streets were indexed.
To evaluate the search engine's performance, addresses from a survey conducted by the NSI were collected, where participants manually provided their addresses. These addresses contain noise: spelling errors, acronyms… Using the national individual code for each participant, the survey data was matched with our national individual-address database, which serves as the ground truth by providing the correct *idAddress*. This annotated dataset, consisting of 100,000 addresses across the country, was used to provide quality metrics on the search engine.
These addresses went through the identification process: - The address was identified 86.0% of the time with an accuracy of 0.94.
- The street was identified 99.5% of the time with an accuracy of 0.94.
- 0.5% of the addresses were not found, at any level. A test set of 100,000 clean addresses, sampled from the data in the address index, was created to evaluate the quality of its own data identification: the address was identified 100.0% of the time with an accuracy of 1 (all with the strict identification step).
To assess the efficiency of the actual identification process, a comparison was made between two different pipelines:
- P1: The actual full pipeline including strict address search, street name search, and then number

and suffix search.
- P2: A pipeline that starts with the street name and then the number and suffix search.

To improve processing times, an additional step has been added to P2, creating the P1 process: a strict address search without **tokenization/ngrams**—which is time consuming—, only on addresses in the town required. This step resolves a significant portion of the addresses, around 30% in practical cases. If the database is clean, like administrative addresses, this rate can rise to 100%. With the same test set of 100,000 clean addresses as before, it took **one-sixth the time** to identify them using this strict identification step compared to doing it without, with the same rate of address/street identification and accuracy. Adding this step does not increase processing time if very few addresses are identified, but it saves time if many addresses are found there.

3 Main findings

The primary findings indicate that Elasticsearch is a highly effective tool for optimizing address identification processes. This is much more effective than strict searches in a SQL-style database, which lack of analyzers and do not account for spelling errors and acronyms.
The P1 structure has proven its effectiveness through the evaluation of the quality of identification in an annotated database containing noisy addresses, as well as by comparing its processing time performance with the P2 pipeline.
This solution is scalable and can be adapted to other country datasets by taking into account the specificities of its national addresses with the boosts. This search engine represents a significant advancement in address data processing, offering data control, independency and reliability for official statistics.

Bibliography