Smart Search and Diagnosis Assistant System for Physicians

Mariia Nazarkevych, Oleh Faizulin, Mykhailo Luchkevych

Lviv Polytechnic National University, 12, Stepana Bandery street, Lviv, 79013, Ukraine

Abstract
Making a correct diagnosis is a complex effort. Depending on the specialty, physicians can have to work with a long list of diseases with intersecting, often not obvious symptoms. Despite that patient treatment should be personal, the nature of diseases is rather repetitive. Patients often share common symptoms based on the location, communication network, season of year, etc. Such repetitiveness introduces a value of having an information system that can make suggestion to the physicians based on a set of factors – symptoms, location, season, diagnosis of other patients, etc. The work described below is based on the industry-standard medical datasets and technologies, but it's not limited to a particular dataset, clinic, language, country, or technology. The system itself provides benefits both for physicians, where they become aware of the situation in their area or departments, and hospitals, where they can collect statistics regarding health situation in the given region or other data projection and analyze the info.

Keywords
healthcare, distributed information system, security.

1. Introduction

Healthcare is one of the most important parts of our live. Good healthcare services result in better quality of live, longevity, attractiveness of the country or specific region for migration. Making a correct diagnosis of a patient's disease is crucial to the patient, physician, and healthcare system in general since mistakes can lead to severe consequences or even patient's death. Usually, physicians rely only on their knowledge and patient feedback, but they does not consider situation in their neighborhood, region, and other factors like trending diseases (viruses), etc.

Having information system that allows to consolidate cross-patient information allows physicians to see insights about trending diseases and understand actual trends in real-time. The same system helps hospitals to analyze trends and make proper forecasts and preparation.

This study involved the development of an expert prototype [1].

A system that helps patients diagnose their illness and provide them with appropriate advice.

In addition, the knowledge management used in the system is learned by the expert system. The aim of this study was to find an appropriate language to represent the patient's medical history and current situation in the knowledge base for expert systems to be able to perform an effective consultation. An example of an expert system was developed using CLIPS (C Language Integrated Production System) with a Java interface [2]. The expert system gave good results in the analysis and in the medical cases tested, and the system was able to determine the correct diagnosis.

An expert system solves problems by modeling the human reasoning process and applying specific knowledge and interfaces. Expert systems also use human knowledge to solve problems that normally require human intelligence. These expert systems represent expert knowledge as data or rules in a computer. These rules and data can be used when needed to solve a problem. A person must read and interpret the knowledge that is used. A computer program designed to simulate the ability of a human expert to solve problems [3].

MoMLeT+DS 2023: 5th International Workshop on Modern Machine Learning Technologies and Data Science, June, 3-4, 2023, Lviv-Leiden, Ukraine
EMAIL: mariia.a.nazarkevych@lpnu.ua (M. Nazarkevych); oleh.r.faizulin@lpnu.ua (O. Faizulin); mykhailo.m.luchkevych@lpnu.ua (M. Luchkevych)
ORCID: 0000-0002-6528-9867 (M. Nazarkevych); 0000-0001-5781-0600 (O. Faizulin); 0000-0002-2196-252X (M. Luchkevych)

Expert systems and artificial intelligence covers such diverse activities as game automated reasoning, natural language, automatic programming, machine learning, robotics and vision, software tools, modeling human performance and expert systems for complex decision making. Comprehensive medical solutions take center stage and every step and our system helps solve this problem with an expert system.

There are many medical diagnostic expert systems MYCIN [4], EasyDiagnosis [5], PERFEX [6], INTERNIST-I [7], ONCOCIN [8], DXplain [9] and PUFF.

MYCIN was the first known medical expert system developed by Shortliffe at Stanford to assist physicians rather than antimicrobial experts. A limitation of MYCIN was that its knowledge base is incomplete because it does not cover the full spectrum of infectious diseases. Running it would require more computing power than most hospitals at the time could afford. Doctors don't like to type in a terminal and demand a much better user interface than the one provided.

EasyDiagnosis is expert system software that lists and clinically describes the most likely conditions based on an analysis of your particular symptoms. EasyDiagnosis focuses on the most common medical complaints for most doctor visits and hospitalizations [10].

2. Goals and Non-goals

The system designed at the article is a distributed search system targeted both for physicians and hospital's analytics department. At the same time, the system is designed to work with personal data, so it should comply to international healthcare and privacy standards [11].

System Requirements

The system should comply to the following functional and non-functional requirements [12]:

- Operate standard medical vocabularies like ICD-10, ICD-11, SNOMED, etc.
- Enable support for localization, including both regional languages and medical vocabularies
- Provide search suggestions to physicians based on gender, age group, symptoms, location, and other factors.
- Provide analytical information based on set diagnosis including various projections like age group, location, etc.
- Comply with industry standards and security requirements like HIPAA, HITECH, GDPR, etc.
- Save cost of implementation and ownership by re-using industry best practices, technologies, and tools.
- Be cloud native yet allow operation on premises.
- Allow seamless integration to the medical software via API.
- 24/7 operations.
- 0-downtime changes to the system dataset.
- Inclusion of diagnostic tooling for system monitoring
- Be SLA/SLO compliant

Non-goals of the system

It must be clearly stated that the system is an assistant to the physician [13]. It's up to physician to make the final decision on diagnosis and treatment. Any system suggestions must be analyzed by responsible physician prior to accepting the suggestion and communicating it to the patient.

Key concepts & terminology used.

During the article the following terms and keywords will be repeated multiple times [14]:

- Indexing – a process of saving and analyzing records into the search index
- Search Index – a database containing analyzed records used for the search
- Analysis – a process of discovering useful data from the raw data
- Stemming – a process of extracting roots from words
- Stop words – a list of words that are filtered out
- Search Rank – a position in search result for the specific query

- Search Score – a single value that represents how close result is matching search query
- Boosting Search Score – a process of applying multiple variables and functions that result to the change of the Search Score
- Query Expansion – a process of reformulation a users' search query to retrieve more relevant results

Competitors

The system is designed to replace in daily use both public solutions and in-house competitors. Due to an NDA, it's not allowed to name specific vendors or systems. The further comparison will be done in the following way [15]:

- System – the system being described in the article.
- Public – systems that are available on the internet and free to use.
- Proprietary – the systems that are available as a separate or a part of a proprietary software solutions.

3. Methods

The component diagram below describes how the application looks like on high-level view [16]. For the sake of simplicity, reference architecture is designed to be run on AWS cloud [17]. At the same time, there are no blockers to host the same software on premises, on Azure cloud or any other hosting platform. Certain specific components like Aurora or OpenSearch [18] are interchangeable with their open-source equivalents like MySQL (PostgreSQL) and Elasticsearch.

Theoretically, it should be possible to use a NoSQL database like MongoDB and other search engines like Solr or Sphinx, however those components were not evaluated for the system feasibility and most likely will require certain changes to the codebase.

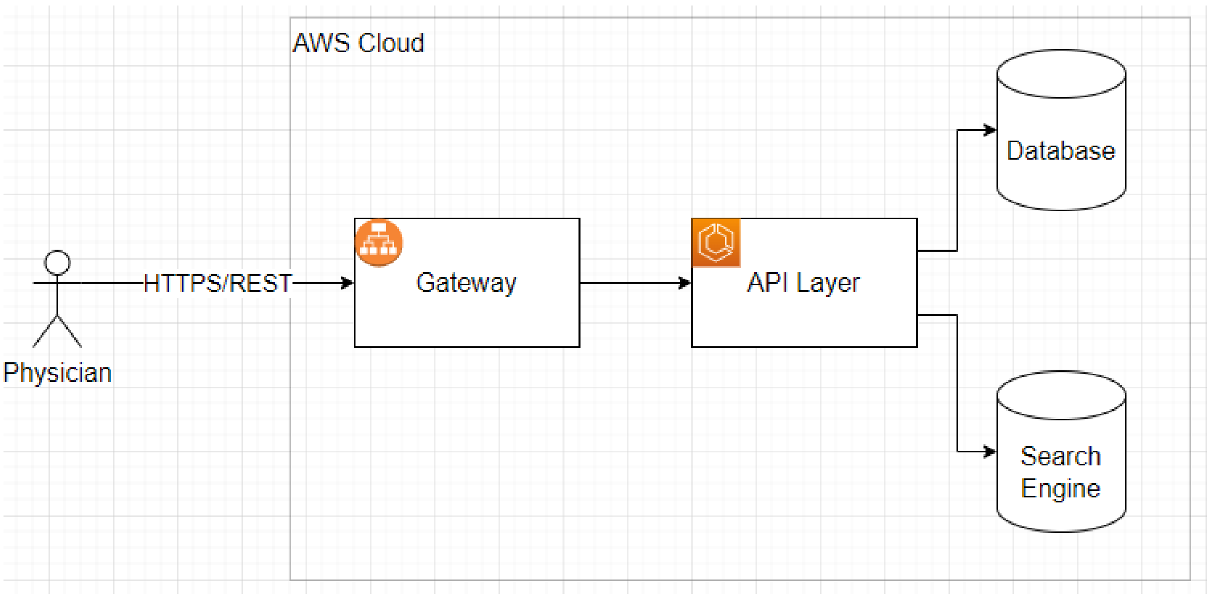


Figure 1: Component view

The Diagram Below illustrates data flow (via action/sequence diagram) of a typical search request

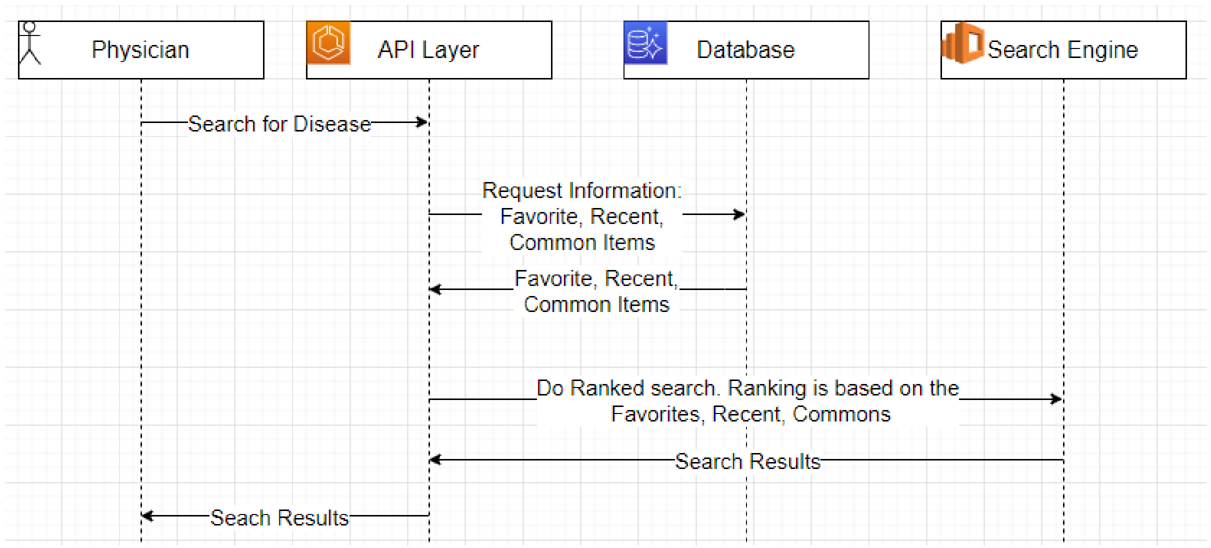

Physician
API Layer
Database
Search Engine
Search for Disease
Request Information: Favorite, Recent, Common Items
Favorite, Recent, Common Items
Do Ranked search. Ranking is based on the Favorites, Recent, Commons
Search Results
Seach Results

Figure 2: A typical search request flow

Whenever physician selects an item on UI, a feedback action is required to update usage statistics which is essential for the system functionality in terms of user experience and statistics

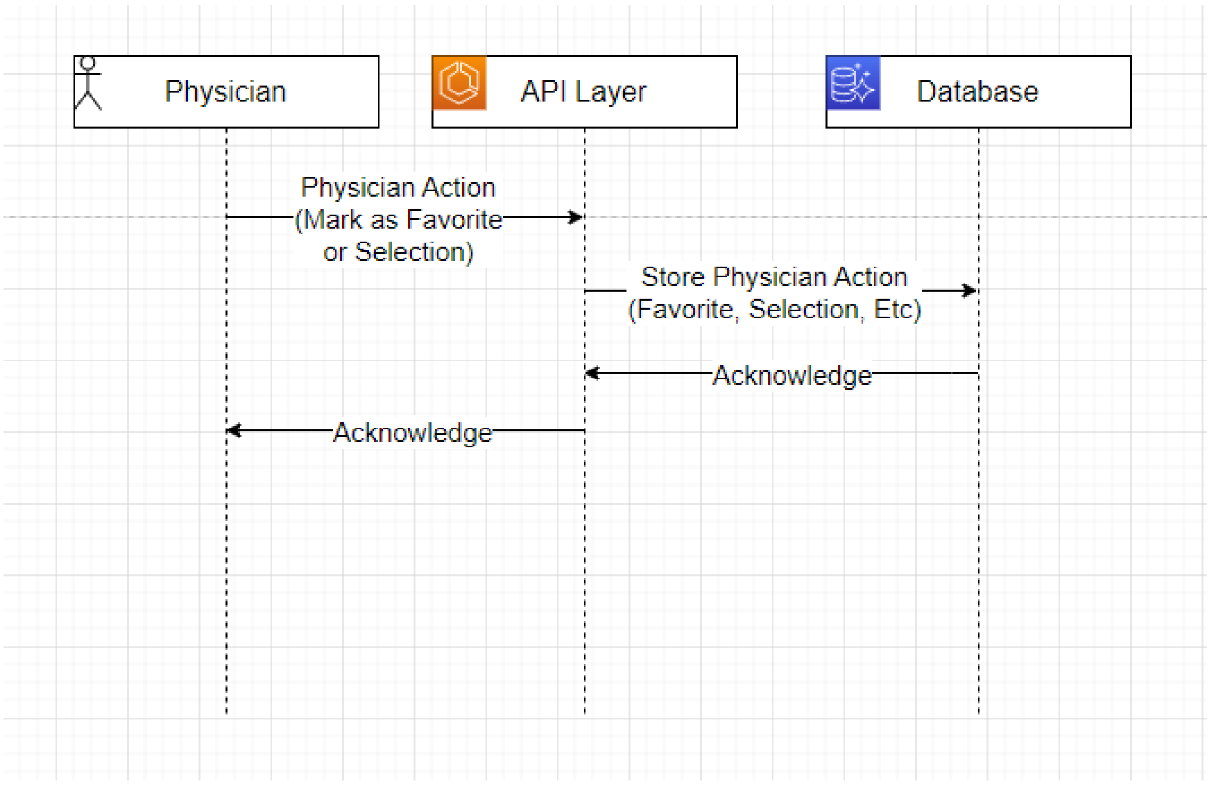

Physician
API Layer
Database
Physician Action (Mark as Favorite or Selection)
Store Physician Action (Favorite, Selection, Etc)
Acknowledge
Acknowledge

Figure 3: A typical selection feedback flow

Searching for diagnosis

Diagnosis search and suggestion is one of the key components of the system. To be able to provide suggestions, the search engine must be pre-populated with medical vocabulary (dataset). For the illustrative purposes, the system will be populated with the ICD-10 dataset having the following field mappings:

- code – a code used to identify the diagnosis
- text – description of the diagnosis
- symptoms – a list of symptoms relevant for the diagnosis
- facets – a list of categories, where a particular diagnosis is applicable

Below you can find a typical ICD-10 record mapped to the fields used by the system [19, 20].

```
PUT icd10/_doc/R50
{
  "code": "R50",
  "text": "Fever of other and unknown origin.",
  "symptoms": ["high temperature", "sore throat"],
  "facets": ["GENDER:ANY", "AGE_GROUP:ANY"]
}
```

Figure 4: A typical IDC-10 record

At the same time, it's not enough just to map records to the fields [21]. A search engine should be configured to understand field mappings and types [22]. At the same time, since indexing is happening on insert or update, a search engine must be additionally configured with the indexing (field analysis) rules. Having correct indexing rules are crucial both for the search accuracy and performance.

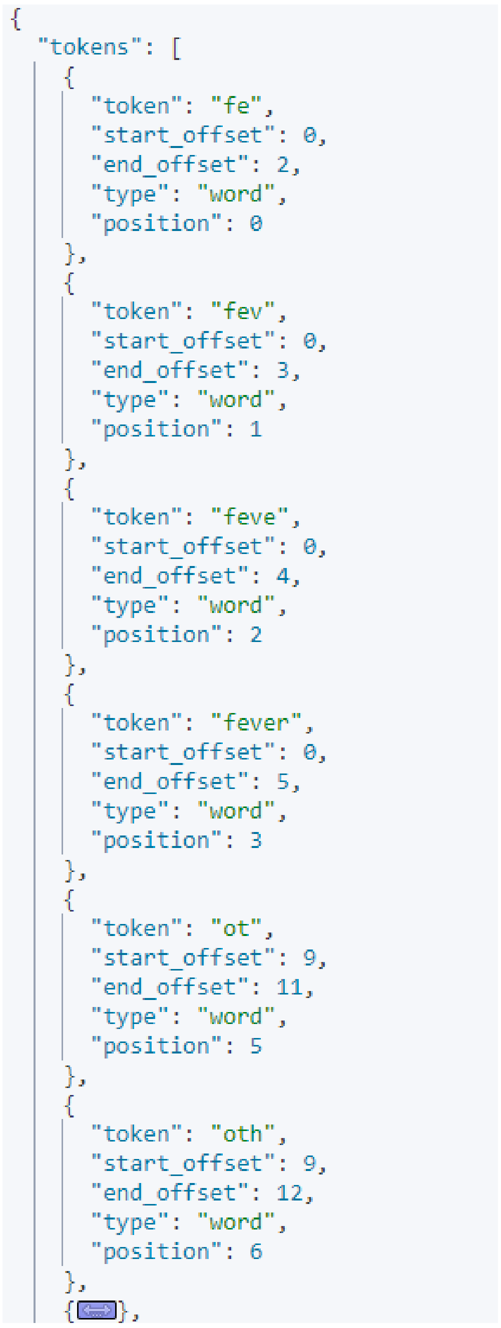
```
PUT icd10
{
  "settings": {
    "analysis": {
      "analyzer": {
        "icd10_analyzer": {
          "filter": ["lowercase", "stop", "snow_eng", "unique"],
          "tokenizer": "en_tokenizer"
        }
      },
      "tokenizer": {
        "en_tokenizer": {
          "type": "edge_ngram",
          "min_gram": 2,
          "max_gram": 10,
          "token_chars": [
            "letter",
            "digit"
          ]
        }
      },
      "filter": {
        "snow_eng": {
          "type": "snowball",
          "language": "English"
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "code": {"type": "text", "analyzer": "icd10_analyzer"},
      "text": {"type": "text", "analyzer": "icd10_analyzer"},
      "symptoms": {"type": "text", "analyzer": "icd10_analyzer"},
      "facets": {"type": "keyword"}
    }
  }
}
```

Figure 5: Reference Elasticsearch schema and analysis (indexing) settings

At the Figure 5 there are two main blocks: “settings” and “mappings”. “Mappings” block is responsible for field types and analyzer selection, while “settings” block is responsible for analysis configuration. Whenever record is inserted into the search engine, it’s being indexed according to the settings (show **Figure 5**). In the current use-case, Elasticsearch is configured to do the following analysis operations on the records inserted (apart from facets fields):

1. Convert text to lowercase
2. Remove stop words
3. Use stemmer
4. Remove duplicated tokens

Once the data is analyzed, it’s being stored in the search index using edge-ngram tokenizer which is configured to store tokens of 2 and up to 10 characters. Such approach allows to implement search-as-you-type style suggestions. For example, let’s analyze the text of R50 diagnosis: **Fever of other and unknown origin.**

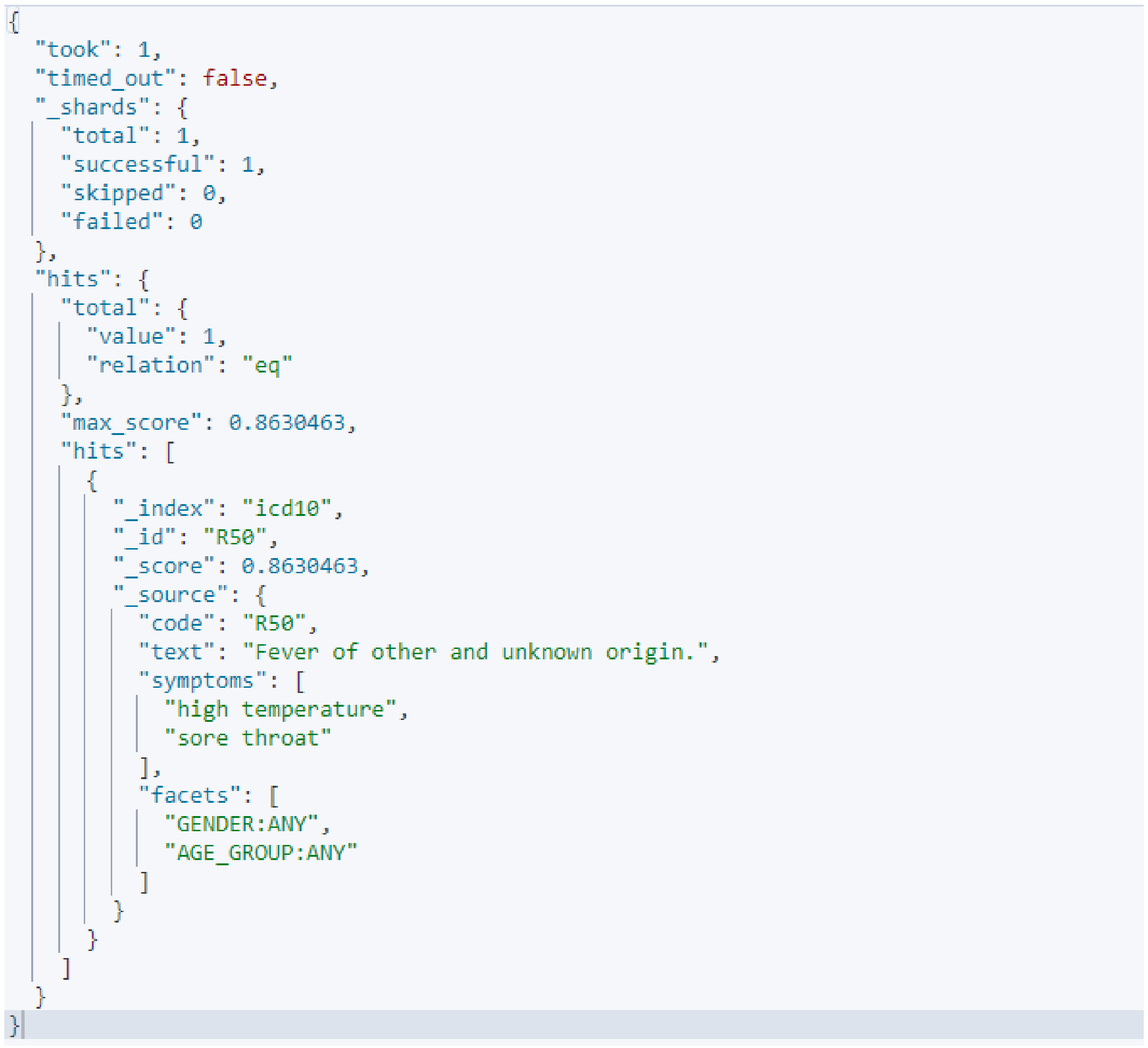


```
{
  "tokens": [
    {
      "token": "fe",
      "start_offset": 0,
      "end_offset": 2,
      "type": "word",
      "position": 0
    },
    {
      "token": "fev",
      "start_offset": 0,
      "end_offset": 3,
      "type": "word",
      "position": 1
    },
    {
      "token": "feve",
      "start_offset": 0,
      "end_offset": 4,
      "type": "word",
      "position": 2
    },
    {
      "token": "fever",
      "start_offset": 0,
      "end_offset": 5,
      "type": "word",
      "position": 3
    },
    {
      "token": "ot",
      "start_offset": 9,
      "end_offset": 11,
      "type": "word",
      "position": 5
    },
    {
      "token": "oth",
      "start_offset": 9,
      "end_offset": 12,
      "type": "word",
      "position": 6
    },
    {},
```

Figure 6: A typical analysis result

As it’s demonstrated on Figure 6, the diagnosis text is split to a set of tokens like “fe”, “fev”, “feve”, etc. These tokens are being used to match the search input and to provide the most relevant results. At the same time, stop words, punctuation symbols and duplicated tokens are removed.

Having the above index, it’s easy to make a search having either full or partial input of the term. The search for user input “feve” would result in item:



```
{
  "took": 1,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 1,
      "relation": "eq"
    },
    "max_score": 0.8630463,
    "hits": [
      {
        "_index": "icd10",
        "_id": "R50",
        "_score": 0.8630463,
        "_source": {
          "code": "R50",
          "text": "Fever of other and unknown origin.",
          "symptoms": [
            "high temperature",
            "sore throat"
          ],
          "facets": [
            "GENDER:ANY",
            "AGE_GROUP:ANY"
          ]
        }
      }
    ]
  }
}
```

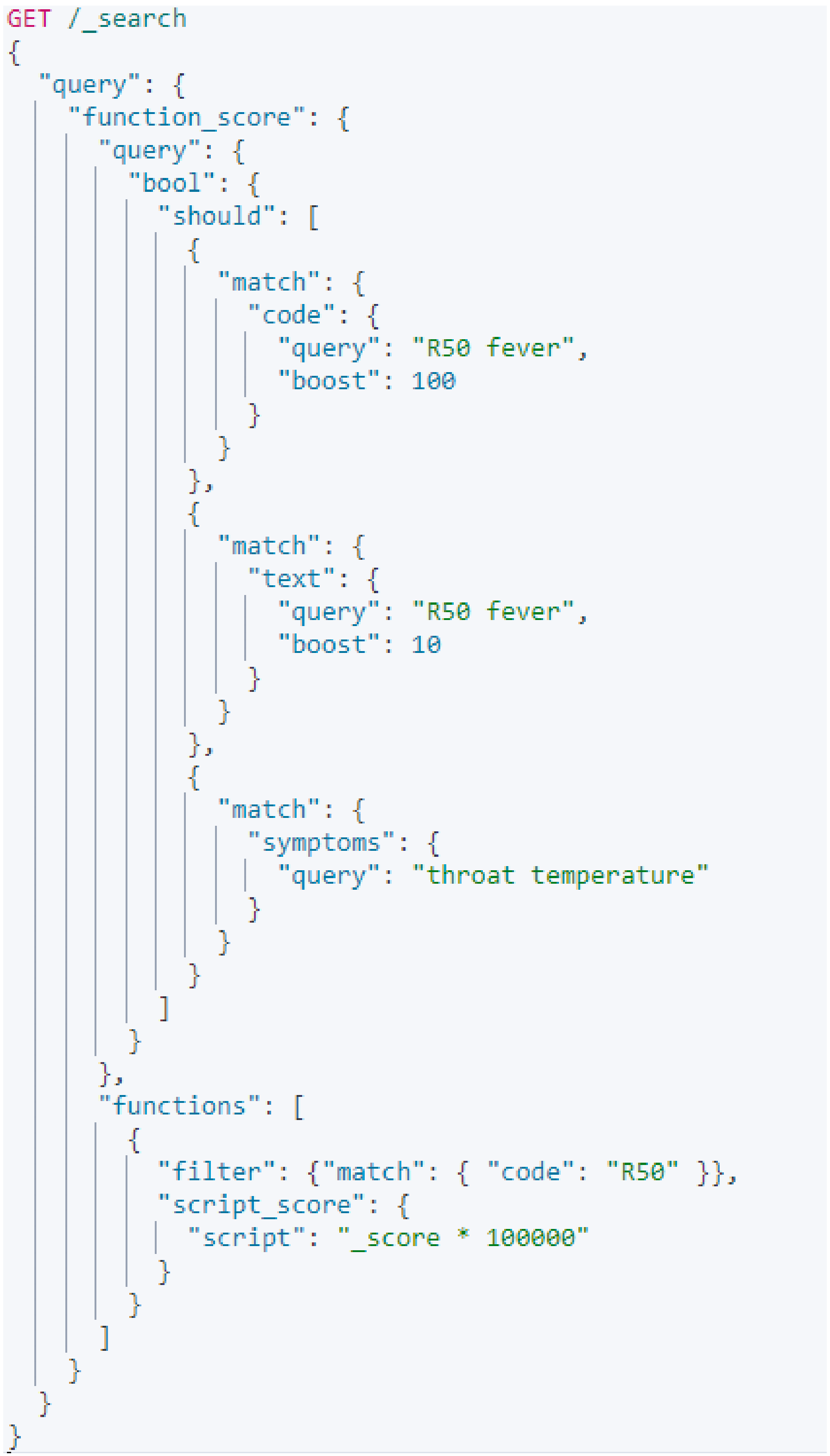
Figure 7: Reference Elastic search schema and analysis (indexing) settings

Additionally, by applying “fuzziness” factor to the query a search behavior can be altered to support a certain number of mistakes. For instance, if “fuzziness: 1” is specified in the search query the system would provide results both for “feve” and “fevr” search input.

Changing ranking of the search results

Being able to re-order search results is crucial for the system to provide a high-quality search result. The system should score items differently if a search input is found in code, text, or keyword field. Typically, the code match must be ranked higher and symptom match must be ranked lower then description field. To achieve such results, the boosting functions are applied.

The similar technique is used when the system wants to suggest an item that is frequently used by other physicians. For instance, if a particular item must be ranked higher a script-based filter function is applied which can manipulate the final search score of the item.



```
GET /_search
{
  "query": {
    "function_score": {
      "query": {
        "bool": {
          "should": [
            {
              "match": {
                "code": {
                  "query": "R50 fever",
                  "boost": 100
                }
              }
            },
            {
              "match": {
                "text": {
                  "query": "R50 fever",
                  "boost": 10
                }
              }
            },
            {
              "match": {
                "symptoms": {
                  "query": "throat temperature"
                }
              }
            }
          ]
        }
      },
      "functions": [
        {
          "filter": {"match": { "code": "R50" }},
          "script_score": {
            "script": "_score * 100000"
          }
        }
      ]
    }
  }
}
```

Figure 8: Boosting of the search result

Localization, Internationalization and Multi-Dictionary support

Since different countries naturally use their own languages and localized versions of medical diseases classification, it’s crucial for the system to support multiple languages. The same applies for a revision of medical classification systems which can add new records once new diseases appear. A good example is COVID-19, which was not initially present in the medical classification systems.

To add support for the custom language and vocabulary, a simple three steps must be taken:

1. A local dictionary must be loaded, for instance ICD-10-FR for France
2. Analysis settings (**Figure 5**) must be adjusted to use French version of the snowball filter
3. A separate index must be used (for instance icd10fr)

Data Updates

In case there is a need to update the existing dictionary, it's simply enough to re-insert data into the existing index. The system will be fully operational during the system update. Physicians might experience a slight slowdown of the system, but typically it's not the case. The data consistency is guaranteed by re-using the same disease codes. The records in the search engine and database may be added or updated, but not removed. The removal of the items is strictly forbidden because it will lead the system to an inconsistent state. Instead, certain items may be marked as excluded for the search.

System Evolution

It must be noted that the system is designed to have 24\7 operations. Eventually, during system evolution there is a need to introduce braking changes to the system. In order to achieve 0-downtime, the changes must be applied in the following order:

- A database or search engine or and API is modified with a backward-compatible change, e.g. a new field, property or endpoint is added
- Consuming components are updated to use a new field, property or API endpoint instead of a legacy one
- The legacy field. Property or API endpoint is removed

Performance & Scaling

System performance heavily depends on multiple factors like hardware, number users, data size, etc. At the same time, it's crucial to follow the following key principles which will heavily impact scalability:

- API layer should be stateless to allow horizontal scalability
- Search Engine should support data sharding and use separate indexes for each dataset
- Database engine should use separate schema per tenant (hospital or network of hospitals)

If the above characteristics are met the modern 4-core/16 GB RAM AWS instances (or their physical counterparts) can serve thousands of search requests per second.

Reporting

An ability to provide multi-dimensional reports is a last but not least goal of the system. The system itself is not generating reports but it acts as a data source for another reporting system. By combining vocabulary data with the export from the database (preferably due to performance) it's possible to achieve a free-form report generation by a 3rd party reporting system. Another option would be to embed a reporting suite directly into the application however it's not recommended due to the extra cost and overall increased complexity of the system. Depending on the requirements, the variety of reporting tools can be used, such as MicroStrategy (commercial), Kibana or even Microsoft Excel.

Security

Any medical system, as well as the system that works with personal data should obey a set of rules and regulations. There are various rules and regulations across the globe, the most known are GDPR for EU and CCPA for US. Besides mentioned rulesets, there is a set of medical regulations such as HIPAA and HITECH which must be considered.

The system achieves security by combining a set of best practices (OWASP Top 10) with the set of special measures applied at the individual layers of the system. Let's look on the layers:

- Search engine is a layer does not require any special treatment because it contains dictionary data.
- Database is the place where all the selected diagnosis are stored. Ideally the database should not contain any PHII (personal health identifiable information) to minimize security risks
- API Layer, should follow industry best practices, but does not require any special treatment

All the communication should be secured and use proper credentials as well as the encryption-in-transit approach.

4. Comparison with competitors

To understand how system performs, let’s do a set of comparisons with the competitors:

Table 1:
System characteristics comparison:

Characteristics	The System	Public	Proprietary
24/7 Operations	Yes	Not Reliable	Requires restart to re-index data or tune search results
Exposes Search API	Yes	No	Yes
Custom Ranking	Yes	No	No
Reporting	Via 3rd party system	No	No
On-Premises hosting	Yes	No	It Depends
Cloud-based hosting	Yes	Yes	Yes

Table 2:
Seach types and time to get results

Seach Type	The System	Public	Proprietary
Regular Search	< 1 sec	1-2 sec	< 1 sec
Type-as-you-search	< 1 sec	Partially Supported	< 1 sec
Search by symptoms	< 2 sec	Not Supported	Not Supported
Seach with typos	Yes	Yes	Yes

Table 3:
Localization and Internationalization

Characteristic	The System	Public	Proprietary
Multiple Languages	Same Instance	N/A	Requires a separate customized installation
Multiple Vocabulary Support	Yes	Yes	Yes

Conclusions

1. The smart search systems are valuable both for physicians and hospitals. They allow to enhance physician workflow as well as provide analytics for hospitals.
2. Usage of open-source components allowed to rapidly create a production-ready prototype of the system with the support of a multiple languages, dictionaries, facets. Moreover, the interchangeable nature of components allows to host the system on premises or in the cloud, powered by cloud technologies like Aurora or OpenSearch.
3. The result of the experiments demonstrates clear advantage of the system in the multiple areas in comparison to the existing public and proprietary classification search systems.
4. While being focused on healthcare domain the system can be adapted to the non-healthcare domain and be used, for instance, in retail by re-using similar concepts and approaches.

References

[1] Abu-Naser, S. S., Al-Dahdooh, R., Mushtaha, A., El-Naffar, M. Knowledge management in ESMDA: expert system for medical diagnostic assistance, 2010.

[2] Al-Shawwa, M., Abu-Naser, S. S. Knowledge Based System for Apple Problems Using CLIPS. International Journal of Academic Engineering Research (IJAER),2019, №3(3), 1-11.

[3] Walek, B., & Fajmon, P. A hybrid recommender system for an online store using a fuzzy expert system. Expert Systems with Applications, 2023, 212, 118565.

[4] Anwar, S. M. Expert Systems for Interpretable Decisions in the Clinical Domain. AI in Clinical Medicine: A Practical Guide for Healthcare Professionals, 2023, p.66-72.

[5] Eugenia, L., Giovanni, A., Andrea, M., Rosaria, R. M., Gianni, V., Mara, B., Paolo, P. Mucinous appendiceal adenocarcinoma invading the bladder: not always an easy diagnosis. A case report. Anti-Cancer Drugs, 10-1097.

[6] Zilio, G., Nørgaard, L. S., Gougat-Barbera, C., Hall, M. D., Fronhofer, E. A., Kaltz, O. Travelling with a parasite: the evolution of resistance and dispersal syndromes during experimental range expansion. Proceedings of the Royal Society B, 2023, 290(1990), 20221966.

[7] Shinde, A., Karle, R., & Wabale, N. Disease prediction using mashine learning.

[8] Moradi, M., Besharati, S., Moghaddam, M. M., Fasihi-Ramandi, M., Azad, Z. M., & Mirnejad, R. A concise review on the antinicrobial pertides and their critical activity against inracellular targets of bacteria. Journal of microbiology, biotechnology and food sciences, 2023, 12(4), e6006-e6006.

[9] Lee, P., Greenes, R. A., Kawamoto, K., Fry, E. A. Integration of knowledge resources into applications to enable CDS: Architectural considerations. In Clinical Decision Support and Beyond , 2023, pp. 789-809.

[10] Rashi, T., & Yom-Tov, E. Ethics of Medical Archival Internet Research Data. Journal of Medical Internet Research, 2023, 25, e43754.

[11] Dhasarathan, C., Hasan, M. K., Islam, S., Abdullah, S., Mokhtar, U. A., Javed, A. R., & Goundar, S. COVID-19 health data analysis and personal data preserving: A homomorphic privacy enforcement approach. Computer Communications, 2023, 199, pp.87-97.

[12] Rashid, U., Saddal, M., Farooq, G., Khan, M. A., & Ahmad, N. An SUI-based approach to explore visual search results cluster-graphs. PloS one, 2023, 18(1), e0280400.

[13] Isbell, L. M., Chimowitz, H., Huff, N. R., Liu, G., Kimball, E., & Boudreaux, E. A Qualitative Study of Emergency Physicians' and Nurses' Experiences Caring for Patients With Psychiatric Conditions and/or Substance Use Disorders. Annals of Emergency Medicine, 2023.

[14] Espada, J. P., Martínez, J. S., Rico, I. C., & Sánchez, L. E. V. Extracting keywords of educational texts using a novel mechanism based on linguistic approaches and evolutive graphs. Expert Systems with Applications, 2023, 213, 118842..

[15] Dimand, A. M., Patrucco, A. S., Rodriguez-Plesa, E., Hiriscau, A. M. Social equity in federal contracting during emergencies: A portfolio management perspective. Public Administration Review, 2023.

[16] Kadhim, J. Q., Aljazaery, I. A., & ALRikabi, H. T. S. Enhancement of online education in engineering college based on mobile wireless communication networks and IoT. International Journal of Emerging Technologies in Learning (Online), 2023, 18(1), 176.

[17] Pourmajidi, W., Zhang, L., Steinbacher, J., Erwin, T., & Miranskyy, A. A Reference Architecture for Observability and Compliance of Cloud Native Applications. arXiv preprint arXiv: 2023, 2302.11617.

[18] Ling, E. R., & Ekstedt, M. A threat modeling language for generating attack graphs of substation automation systems. International Journal of Critical Infrastructure Protection, 2023, 100601.

[19] Dronyuk I, Nazarkevych M, Poplavska Z. Gabor filters generalization based on ateb-functions for information security. InInternational Conference on Man–Machine Interactions 2017 Oct 3 pp. 195-206. Springer, Cham.

[20] Nazarkevych, M., Oliarnyk, R., Troyan, O., Nazarkevych, H. Data protection based on encryption using Ateb-functions. In 2016 XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT), 2016, September, pp. 30-32. IEEE.

[21] Medykovsky, M., Droniuk, I., Nazarkevich, M., & Fedevych, O. Modelling the pertubation of traffic based on ateb-functions. In Computer Networks: 20th International Conference, CN 2013, Lwówek Śląski, Poland, June 17-21, 2013. 2013. Proceedings 20 (pp. 38-44). Springer Berlin Heidelberg.

[22] Ahn, J. K., Choi, S., Hasegawa, S., Hayakawa, S. H., Hong, J., Ichikawa, Y., Yoshida, J. Superconducting dipole magnet for Hyperon spectrometer. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2023, 1047, 167775.