|  |  |
| --- | --- |
|  |  |
|  |  |
|  | |
| **Questionnaire Drawing box** | |
| Technology/Model overview | |
| July 2021 | |

Contents

[**RFP/DDQ Automation** 3](#_Toc74763450)

[**Conceptual Analysis:** 3](#_Toc74763451)

[**Process steps** 3](#_Toc74763452)

[**Fig. 1: Flow Chart of RFP/DDQ Automation Model** 4](#_Toc74763453)

[**Assumuption** 5](#_Toc74763454)

[**Tentative timeline** 5](#_Toc74763455)

[**Appendices** 5](#_Toc74763456)

[**References** 5](#_Toc74763457)

# **Job recommendation system**

# **How to make search as smart search (Understanding intents)**

To make search engine as smart search, first intents need to be understood and categorised. A user search intent reveals what someone wants when they search something. User search intent has become the sweetheart of the optimization world, whether for SEO, conversion rate optimization (CRO), or other disciplines because it provides very specific insight without being stalkerish. The user provides the intent, and all companies have to do is pay attention and correctly interpret it. When search engine know *why* customers want something, it can understand *how*to deliver it.

The goal is to extract insights that is as rich and actionable as possible from the queries by predicting the user intent. The question we sought to answer is how can a search engine understand the intent behind a query?

Search engine can be leveraged user intent by using AI/machine learning (ML) model.

**AI/ML model for predicting user intent**

User intent can be broadly classified into four categories. These types of intent correspond to the layers in the marketing funnel:

|  |  |
| --- | --- |
| Informational or Awareness  Related to finding information about a topic.  Examples of informational searches:   * “What is HR support system?” * “What is drawing box” | Transactional  Related to accomplishing a goal or engaging in an activity.  Examples of transactional searches:   * “Take subscription of drawingbox” * “subscription cost” |
| Navigational  Also called “Visit in person.” Related to finding a nearby place or other types of local information.  Examples of navigational searches:   * “Drawing box” * “Login to drawingbox” | **Consideration**  These are in between informational and transactional intent.  Examples of navigational searches:   * “drawingbox reviews” * “drawingbox vs other HR services” |

 To predict user intent using apply ML model, first the queries need to be transformed into their mathematical equivalents. This problem falls in the[natural language processing (NLP](https://en.wikipedia.org/wiki/Natural_language_processing)) category.

The transformation from words into an understandable format can be achieved with the help of ML models such as[BERT, GloVe](https://nlp.stanford.edu/projects/glove/) or[FastText](https://fasttext.cc/). These tools convert each word into a set of numbers (embedding) while simultaneously maintaining the relationship between the words. This means two words that are related (such as “buy” and “shopping”) will be seen as more closely related than two unrelated words (such as “buy” and “laptop”).

Here, working with utterances or queries, it is very natural to structure the model in a recurrent way, rather than treating them independently. This is because of the temporal order of the words. Thus, a linear recurrent neural network model (RNN-1) was the first thing which can be used to model the intent prediction problem.

Then, the complexity of the model is increased, in order to see if adding non-linearities (RNN-2), more complicated recurrent layers, or stacking multiple recurrences (RNN-3) would help solving the problem. In the end, 3 recurrent models can be used, each increasing in complexity compared to the previous one.

A convolutional neural network with no recursion (CNN-1) can also be built, where each word in the query is independent from the previous ones, using max-pooling over time. Conceptually, each word is filtered using a convolutional layer; then for each index in the feature of the words, we compute the maximum value, in the end resulting in a single feature over the entire query.

The input for the neural networks is a word embedding (BERT, GloVe, Fasttext, One-hot), producing a probabilistic output for each of the three intents, from a softmax activation function. Then, during test time, the chosen intent is the one with the highest probability (single-intent prediction), or the probabilities themselves (multi-intent prediction).

Multiple models can be constructed, starting from a very basic set of words, and then improving each method iteratively. While varying the model architecture may bring a very small improvement, using a richer word representation will improve the results by a few percent. User search intent may retains a level of ambiguity because not even humans can agree on the same way to label a specific data set, but search can be made smart by using the NLP and ML models.

<https://backlinko.com/hub/seo/search-intent>

<https://agencyanalytics.com/blog/keyword-search-intent>

<https://www.algolia.com/blog/ai/how-to-identify-user-search-intent-using-ai-and-machine-learning/>

<https://searchengineland.com/deciphering-search-intent-5-areas-to-get-you-started-305550>

<https://gofishdigital.com/query-intent-contextual-histories/>

# **What kind of data will be collected to give best insights? Focus on data recommendations.**

# **How to represent a normal data into meaningful data? What method and pipeline are required?**

Building a system to find and contextualize information to automate the query and respond process. This system can be classified as:

# **How to give personalised experience, how to create context based recommendations?**

Personalization and recommender systems can potentially reduce the omnipresent information overload in our networked world. The integration of context into recommender systems adds an additional dimension of complexity to the recommender data model because ratings may be valid in one particular context only.

Context can be described by a vector of context attributes, e.g. time, location or currently available network bandwidth in a mobile scenario. The actually used context attributes are largely dependent on the application domain. Several principal approaches can be identified to introduce context in different types of recommenders. In a content (or knowledge-based) approach, rules could be used to capture preferences or restrictions arising from context constraints. The problem is to identify relevant rules that can be applied to filter items.

In collaborative filtering, the idea is to associate ratings with a snapshot of the context when the rating is made. A reduction-based approach which eliminates “out of context” ratings. Context to weigh rating according to context similarity is used. Thereby, a major problem is sparseness of available ratings in the same (or comparable) contexts. Another approach is to combine different kind of recommenders to reduce the complexity of the item-user context matrix by applying a cascading hybrid recommender. This means, first only two dimensions of the user-item-context matrix are analyzed, and in a second step the third dimension is considered in addition:

1. Use content- or knowledge based filtering to find relevant items based on context, for example taking the current end user device and location into account

2. Apply collaborative filtering to rank and additionally filter the result set from step 1

**Contextual Information in Recommender Systems**

In our classification of context in recommender systems, we follow many previous approaches by assuming the existence of certain contextual factors, such as time, location, and the purchasing purpose, that identify the context in which recommendations are provided. It can be assumed that each of these contextual factors can have a structure; the Time factor, for example, can be defined in terms of seconds, minutes, hours, days, months, and years. The classification of context that we propose in this article is based on the following two aspects of contextual factors: what a recommender system may know about these contextual factors, and how contextual factors change over time.

One can classify the knowledge of a recommender system about the contextual factors into three categories: fully observable, partially observable, and unobservable.

**Fully observable:** The contextual factors relevant to the application, as well as their structure and their values at the time when recommendations are made, are known explicitly.

**Partially observable:** Only some of the information about the contextual factors, as described above, is known explicitly.

**Unobservable:** No information about contextual factors is explicitly available to the recommender system, and it makes recommendations by utilizing only the latent knowledge of context in an implicit manner.

|  |  |  |  |
| --- | --- | --- | --- |
| How Contextual Factors Change | Fully Observable | Partially Observable | Unobservable |
| Static | Everything Known about Context | Partial and Static Context Knowledge | Latent Knowledge of Context |
| Dynamic | Context Relevance Is Dynamic | Partial and Dynamic Context Knowledge | Nothing Is Known about Context |

How Contextual Factors Change over Time Depending on whether contextual factors change over time or not, we have the following two categories: static and dynamic.

**Static:** The relevant contextual factors and their structure remains the same (stable) over time. For example, in case of recommending a purchase of a certain product, such as a shirt, we can include the contextual factors of Time, Purchasing Purpose, Shopping Companion and only them during the entire lifespan of the purchasing recommendation application.

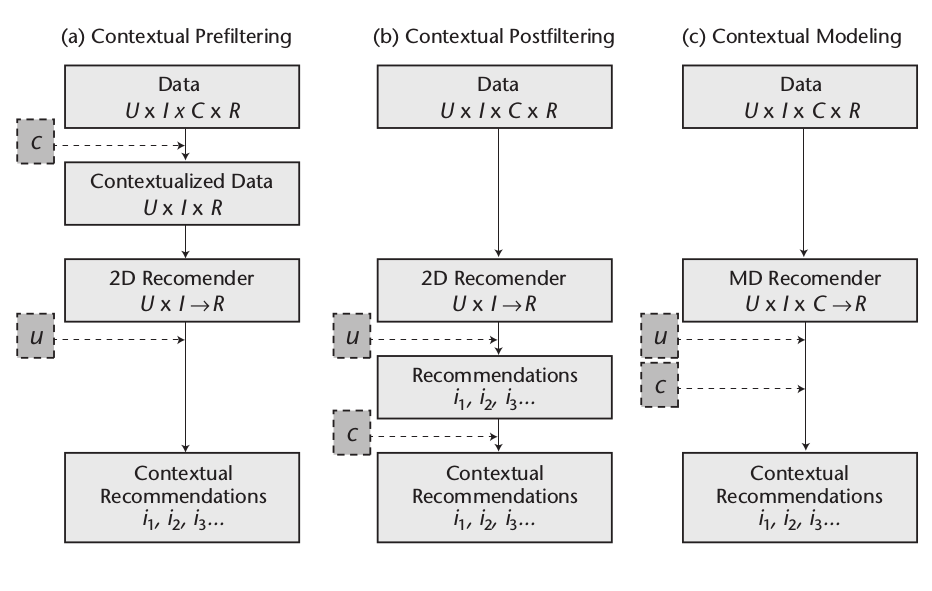
**Dynamic:** This is the case when the contextual factors change in some way. For example, the recommender system (or the system designer) may realize over time that the Shopping Companion factor is no longer relevant for purchasing recommendations and may decide to drop it. Furthermore, the structure of some of the contextual factors can change over time (for example, new categories can be added to the Purchasing Purpose contextual factor over time).

**Context based Recommendation Systems**

Context aware recommender systems are built based on the knowledge of partial contextual user preferences and typically deal with data records of the form <user, item, context, rating>, where each specific record includes not only how much a given user liked a specific item, but also the contextual information in which the item was consumed by this user (for example, context = Sunday). Furthermore, unlike the traditional recommendation process that does not take context into account, the information about the current context***C*** can be used in various stages of the recommendation process, leading to several different approaches to context-aware recommender systems.

In particular, from the algorithmic perspective, while all the context-aware recommendation approaches would work with the data of the form U x I x C x R, where C is an additional contextual dimension, and produce a list of contextual recommendations (i1, i2, i3,...) for each user u, the context-aware recommendation process can take one of the following three forms, based on how the contextual information is used, as shown in figure below:

* **Pre-filtering:** Context is used to select some set of data and then predict like a traditional recommender system. One way of doing that is splitting items or users by context as if they were different items or users.
* **Post-filtering:** Ratings are predicted and then the results are filtered using the context. This can be implemented by ordering the results depending on the context or by just filtering the results.
* **Modeling:** The context is used right in the model. It is more complex and could be implemented by multiple machine learning models like SVM, matrix factorization or a markov chain.



Making combinations of *pre-filtering*, *post-filtering* and *contextual modeling* can also be taken into account.

**Deep learning in Context based Recommendation Systems**

Since context is often represented by dynamic and high-dimensional feature space in multiple applications and services, contextual information can be provided to model in various ways for multiple purposes, such as rating prediction, generating top-k recommendations, and classification of users’ feedback. Specifically, deep context-aware recommendation models based on explicit, unstructured, and structured latent representations of contextual data derived from various contextual dimensions (e.g., time, location, user activity).

Traditional context-aware recommender systems (CARSs) [Adomavicius and Tuzhilin 2015] mostly use predefined explicit contextual information for the recommendation process. The specific contexts describe the circumstances of the information collection, such as weather conditions (sunny, cloudy, raining, etc.) or time conditions (weekday, weekend, etc.). The main advantage of these explicit contexts is their lower dimensionality encompassing only few contextual variables for a specific application domain. However, explicit CARSs have the following limitations:

(1) The selection of specific contexts for the service is a resource-demanding task since it is performed manually by domain experts;

(2) Predefined contexts may not represent the most effective and all-encompassing set of contextual features for the recommendation application;

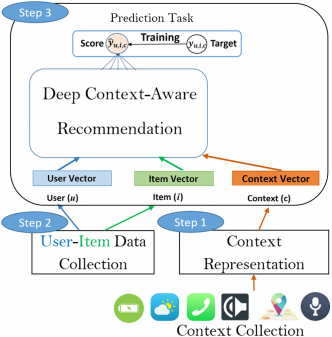
(3) Using explicit contexts, such as location of the customer, may raise privacy issues [Lane et al. 2010] since the exact context is known to the service.

These limitations with explicit contexts may cause serious problems in several practical applications, such as smart health and well-being [Chen et al. 2012], mobile sensing, and Internet of-Things (IoT) [Lane et al. 2015], where the contextual feature space is complex and dynamic.

Deep Learning based CF approaches can be used to overcome these limitations.

**User-Item Data Collection**

Standard process in recommendation systems is depicted in the Figure below, in which users’ feedbacks or ratings are collected for each of the items the system wishes to recommend. Users and items are represented in a standard way as one-hot vectors by their index ID (denoted by “User (u)” and “Item (i)” in Figure below)

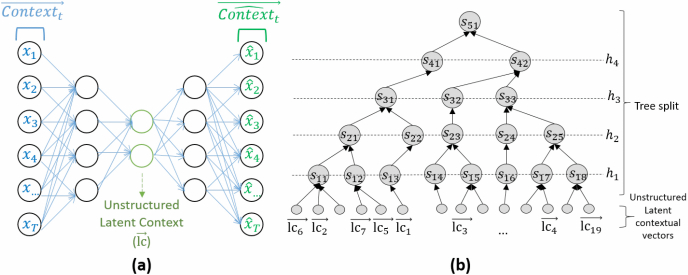


Deep context-aware recommendation framework

**Context representations**

Context can be represented in three different ways: explicit, latent unstructured, and latent structured. For each of the context representations, first normalize the contextual feature values to a scale of 0 to 1 and transform nominal features to binary features. Extracting explicit contextual representation includes using all of the available (raw) T contextual features, whereas the unstructured latent contextual information is a compact representation containing L latent values (L < T) extracted from the hidden layer of an AE [Unger et al. 2016], as shown in Figure 2(a). To automatically learn the structure of latent contextual variables and the semantically meaningful interrelationships among them.

A structured latent contextual representation for the recommendation process [Unger and Tuzhilin 2019] is also utilized, which includes a set of contextual situations (clusters IDs) at different granularity levels extracted from a hierarchical tree, as shown in Figure 2(b)

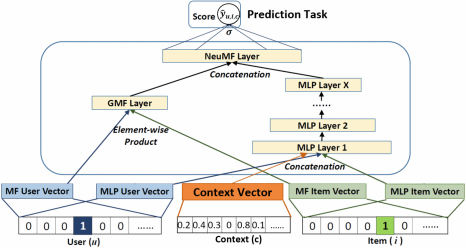


(a) Unstructured latent context [Unger et al. 2016]. (b) Hierarchical latent context.

**Deep Context-Aware Recommendation**

The recommendation algorithm can be any CF recommendation model that takes into account user, item, and context representations for the prediction task.

Specifically, a contextual vector (denoted by “Context (c)” in Figure 4) that is modeled as a set of explicit, unstructured, or structured latent contextual features. The contextual vector c is concatenated to the user *u* and item *i* embedding to learn a new nonlinear function between all three components (users, items, and contexts). In this way, the dimension of context is considered within the neural framework to automatically learn its influence on the overall predicted target.



**Figure: Neural-MF (NeuMF) extension with contextual information**

In the case of rating prediction task (as shown as an example in Figure 1), the target is a predicted rating score. In case of the classification task, softmax activation function can be applied that provides the probability of each class label (categorical value), and then we predict the class with the highest probability. The NeuMF model that is shown in Figure 4 is a generalization of the NCF model with new learned interactions between user, item, and context features. It consists of two major components: generalized matrix factorization (GMF) and multilayer perceptron (MLP). The GMF (denoted by “GMF Layer” in Figure 4) uses a linear function that is learned by embedding of one-hot vectors of users and items. The second component is the MLP, which can endow the model with a large amount of flexibility and nonlinearity to learn the interactions between user, item, and contextual vector representations (explicit, structured, and unstructured latent contexts).

**Conclusion**

* Context-aware methods used for multiple tasks, such as rating prediction, generating top-k recommendations, and classification of users’ feedback.
* Utilizing structured latent contexts in the deep recommendation framework achieves significantly better performance than the other context-aware models across all tasks.
* For high-dimensional contexts, a relatively small size of contextual situations is what one actually needs for the recommendation process, whereas for low-dimensional contexts, it is preferable to expand the number of possible contexts for the recommendation process.

**Conceptual Analysis:**

Building a system to find and contextualize information to automate the query and respond process. This system can be classified as:

1. Generative and retrieval systems,
2. Single-round question answering,
3. Multi-round question answering systems,
4. Open question answering systems, and
5. Specific question answering systems.

With the application of deep learning in natural language processing (NLP), machine reading can automatically find answers to match questions directly from the documents. The deep learning language model converts the questions and documents to semantic vectors (embedding) to find the matching answer.

**Process steps**

The flow chart of RFP generator is depicted in fig.1, the steps involved in RFP generator are briefly explained as follow:

1. **Create Database:** The custom training dataset created by domain experts from Delivery (in this case FMS). This dataset need to be preprocessed and standardized before passing to the embedding model.
2. **Embedding model:** Various deep learning models which can be used to generate text embedding such word2vec, doc2vec, node2vec and sentence2vec to convert these questions into feature vectors and store them in database and would assign a vector ID to each feature vector at the same time.
3. **Semantic analysis:** The user inputs a query, and after generating the feature vector through embedding model, one can find the most similar query in the embedding database. There are various methods to find the similarity between two vectors such as cosine distance is used to represent the similarity between two sentences. Because all vectors are normalized, the closer the cosine distance of the two feature vectors to 1, the higher the similarity. In practice, the system may not have perfectly matched questions in the library. Then, one can set a threshold of 0.9. If the greatest similarity distance retrieved is less than this threshold, the system will prompt that it does not include related query.

**Fig. 1: Flow Chart of RFP/DDQ Automation Model**

|  |  |
| --- | --- |
| **Create Database** |  |
|  |  |
| **Train embedding model** |  |
|  |  |
| **Generate** **embedding** |  |
|  |  |
| No  **Save the model?** |  |
| Yes | **Save trained model with embedding** |
|  |  |
| **Query** |  |
| **Generate embedding using trained model** |  |
|  | **Upload Model** |
|  |  |
| **Semantic analysis using NLP Technique** |  |
|  |  |
| No Yes  **Predicting the answer?** | **Display the response of query** |

**Assumuption**

1. Training dataset will be provided by delivery team in July 1st week 2021
2. We can expect few changes to the training dataset based on the intial review of the data shared by delivery
3. API Integration will be done between Client instance and Alexandria Instance (where ML model saved/run)
4. Model Testing will be done by delivery (offline during the intial stages of the project)

**Tentative timeline**

|  |  |
| --- | --- |
| Training dataset | Around July 1st 2021 |
| Finalizing the plan of action | Around July 12th 2021 |
| Beta version of the Model | August 31st 2021 |

**Appendices**

RFP (Request for Proposal): As institutional investors have large pool of experts to service their request, they tend to choose the preferred partner by RFP responses evaluate the merits of each vendor compared to others

DDQ (Due Deligence Questions): As institutional investors increase their focus on issues related to alignment of interest, governance and transparency with their private equity manager relationships, the level of detail required for their upfront fund diligence process has increased. This increase has resulted in the proliferation of lengthy, customized due diligence questionnaires (“DDQ”) by many Limited Partners, advisors and placement agents. These customized DDQs, which have varying content and length, have created an extraordinary administrative burden on all interested parties, including Limited Partners, General Partners and Placement Agents.

**References**

<https://rfp360.com/rfi-rfp-rfq/>

<https://ilpa.org/wp-content/uploads/2018/09/ILPA_Due_Diligence_Questionnaire_v1.2.pdf>

<https://www.microsoft.com/en-us/research/publication/mcr/>

<https://www.tensorflow.org/lite/examples/bert_qa/overview>

<https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15812785.pdf>

<https://huggingface.co/transformers/examples.html>

<https://medium.com/analytics-vidhya/semantic-similarity-in-sentences-and-bert-e8d34f5a4677>

<https://medium.com/saarthi-ai/build-a-smart-question-answering-system-with-fine-tuned-bert-b586e4cfa5f5>

<https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>

<https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>

NOTWITHSTANDING ANYTHING CONTAINED IN THIS DOCUMENT TO THE CONTRARY, ACUITY KNOWLEDGE PARTNERS IS NOT AGREEING TO ANY LEGAL OR CONTRACTUAL TERMS, CONDITIONS, OR OBLIGATIONS IN CONNECTION (AS APPLICABLE). ACUITY KNOWLEDGE PARTNERS EXPRESSLY RESERVES THE RIGHT TO FULLY AND FREELY NEGOTIATE ALL RELEVANT LEGAL TERMS (AS APPLICABLE).

ALL INFORMATION CONTAINED HEREIN IS PROTECTED BY LAW AND NONE OF SUCH INFORMATION MAY BE COPIED OR OTHERWISE REPRODUCED, REPACKAGED, FURTHER TRANSMITTED, TRANSFERRED, DISSEMINATED, REDISTRIBUTED OR RESOLD, OR STORED FOR SUBSEQUENT USE FOR ANY PURPOSE WHATSOEVER, IN WHOLE OR IN PART, IN ANY FORM OR MANNER OR BY ANY MEANS WHATSOEVER, BY ANY PERSON WITHOUT ACUITY KNOWLEDGE PARTNERS PRIOR WRITTEN CONSENT.

ALL INFORMATION CONTAINED HEREIN IS BELIEVED TO BE ACCURATE AND RELIABLE. BECAUSE OF THE POSSIBILITY OF HUMAN OR MECHANICAL ERROR AS WELL AS OTHER FACTORS, HOWEVER, ALL INFORMATION CONTAINED HEREIN IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND.

TO THE EXTENT PERMITTED BY LAW, ACUITY KNOWLEDGE PARTNERS AND ITS DIRECTORS, OFFICERS, EMPLOYEES, AGENTS, REPRESENTATIVES, LICENSORS AND SUPPLIERS DISCLAIM LIABILITY TO ANY PERSON OR ENTITY FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL LOSSES OR DAMAGES WHATSOEVER ARISING FROM OR IN CONNECTION WITH THE INFORMATION CONTAINED HEREIN OR THE USE OF OR INABILITY TO USE ANY SUCH INFORMATION, EVEN IF ACUITY KNOWLEDGE PARTNERS OR ANY OF ITS DIRECTORS, OFFICERS, EMPLOYEES, AGENTS, REPRESENTATIVES, LICENSORS OR SUPPLIERS IS ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSSES OR DAMAGES.

TO THE EXTENT PERMITTED BY LAW, ACUITY KNOWLEDGE PARTNERS AND ITS DIRECTORS, OFFICERS, EMPLOYEES, AGENTS, REPRESENTATIVES, LICENSORS AND SUPPLIERS DISCLAIM LIABILITY FOR ANY DIRECT OR COMPENSATORY LOSSES OR DAMAGES CAUSED TO ANY PERSON OR ENTITY, INCLUDING BUT NOT LIMITED TO BY ANY NEGLIGENCE (BUT EXCLUDING FRAUD OR ANY OTHER TYPE OF LIABILITY THAT, FOR THE AVOIDANCE OF DOUBT, BY LAW CANNOT BE EXCLUDED) ON THE PART OF, OR ANY CONTINGENCY WITHIN OR BEYOND THE CONTROL OF, ACUITY KNOWLEDGE PARTNERS OR ANY OF ITS DIRECTORS, OFFICERS, EMPLOYEES, AGENTS, REPRESENTATIVES, LICENSORS OR SUPPLIERS, ARISING FROM OR IN CONNECTION WITH THE INFORMATION CONTAINED HEREIN OR THE USE OF OR INABILITY TO USE ANY SUCH INFORMATION.

WITH RESPECT TO THE INFORMATION PROVIDED HEREIN, ACUITY KNOWLEDGE PARTNERS HEREBY DISCLAIMS ANY WARRANTY, EXPRESS OR IMPLIED IN RELATION TO FITNESS FOR ANY PARTICULAR PURPOSE.