Enriching Word Embeddings with Food Knowledge for Ingredient Retrieval

Álvaro Mendes Samagaio
Faculty of Engineering, University of Porto, Portugal
Fraunhofer Portugal, Porto, Portugal

Henrique Lopes Cardoso
Faculty of Engineering, University of Porto, Portugal
Artificial Intelligence and Computer Science Laboratory (LIACC), Porto, Portugal

David Ribeiro
Fraunhofer Portugal, Porto, Portugal

Abstract

Smart assistants and recommender systems must deal with lots of information coming from different sources and having different formats. This is more frequent in text data, which presents increased variability and complexity, and is rather common for conversational assistants or chatbots. Moreover, this issue is very evident in the food and nutrition lexicon, where the semantics present increased variability, namely due to hypernyms and hyponyms. This work describes the creation of a set of word embeddings based on the incorporation of information from a food thesaurus – *LanguaL* – through retrofitting. The ingredients were classified according to three different facet label groups. Retrofitted embeddings seem to properly encode food-specific knowledge, as shown by an increase on accuracy as compared to generic embeddings (+23%, +10% and +31% per group). Moreover, a weighing mechanism based on TF-IDF was applied to embedding creation before retrofitting, also bringing an increase on accuracy (+5%, +9% and +5% per group). Finally, the approach has been tested with human users in an ingredient retrieval exercise, showing very positive evaluation (77.3% of the volunteer testers preferred this method over a string-based matching algorithm).

2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Computing methodologies → Knowledge representation and reasoning; Computing methodologies → Lexical semantics

Keywords and phrases Word embeddings, Retrofitting, LanguaL, Food Embeddings, Knowledge Graph

Digital Object Identifier 10.4230/OASIcs.LDK.2021.15

Funding *Henrique Lopes Cardoso*: This research is partially supported by LIACC (FCT/UID/CEC/0027/2020), funded by Fundação para a Ciência e a Tecnologia (FCT).

1 Introduction

Conversational agents and smart assistants are an interesting opportunity for many application areas [32]. Fostered by the latest advances in artificial intelligence and natural language processing [12], these software allow interaction with computer systems through conversation or chat interfaces [6]. From an interaction point of view, they enable intuitive interaction to access different services. Conversational agents are also cost-effective and may, in may cases, replace human labour [13] in providing access to services from simple access to information to more complex services including infotainment, customer support [34], and recommendation systems. Smart assistants may also positively impact user health by interfacing with health-related services [4, 1].

In the context of food and nutrition, conversational agents may interface with systems that help its users acquiring healthier eating habits, inform and support their decisions [8] which may help preventing several chronic diseases [9]. One of the challenges in developing conversational agents for this domain is related to the complexity of the food taxonomy, which is rich in synonyms, hypernyms and hyponyms [16, 30]. For example, when users refer to *cheese*, they are mentioning a large group of different types of cheese and any of them could be considered. This work could be very useful for tasks where the system must match different sources of information. In this case the goal was to match a query in the form of a named entity with all the entries in a database.

In this work, we describe an approach that exploits semantic knowledge from the food domain in a food matching algorithm. More specifically, we explore the enrichment of pre-trained word embeddings with food-domain-specific knowledge, which can then be used in a number of tasks. To the best of our knowledge, this is the first set of word vectors that truly incorporate semantic information from a knowledge graph focused in food-related concepts.

2 Related Work

This section explores the work that has been done and is currently applied regarding word representations and semantic knowledge.

There are several works that study word representations in high dimensional spaces, mainly focusing on capturing context from large corpora. The underlying premise is that contextual information constitutes a proper representation of linguistic items. Word representations have gained increased attention in NLP tasks with the work of Mikolov et al. [17] (Word2Vec). Other sets of pre-trained embeddings use slightly different techniques to capture and encode the semantic and contextual information in texts. *GloVe* [19] is trained on global word-word co-ocurrence statistics aggregated from a corpus. It uses a log-bilinear regression model to create the word vectors, hence combining the features that come from global matrix factorization and also from local context windows.

It is possible to find pre-trained lexicons created through the application of Word2Vec or GloVe-like algorithms to huge corpora (such as the Google News Dataset). These lexicons are generic enough to capture the meanings of the words. However, this can be seen as one of their greatest disadvantages, which is the fact that they were not trained specifically for a given domain, and thus may not represent well enough the semantics of that domain. More recently, the development of Language Models that make use of transformer architectures [31], such as BERT [7] or ELMo [20], shifted the state-of-the-art on word representations for NLP, from the pre-trained and static embeddings to contextualized embeddings. These models use self-attention layers that change the embeddings of each word according to its context. In this case, the same word in different contexts will be represented by different vectors.

Besides word vector representations, there are other resources that can be used to portray concepts and their relations. Examples include knowledge graphs or ontologies that encode semantic relations in a graph which connects concepts that are linked in some way. There is a wide range of knowledge graphs available and they can also be used by conversational agents to retrieve information according to a given query [5, 3, 4]. It is known that commercially available smart assistants such as *Google Assistant* and *Siri* use knowledge graphs to process the information inputted by the user in order to retrieve the correct answers [15]. One of the most well-known knowledge graphs is *WordNet* [18], a lexical database for the English language, that also includes synonyms and definitions for words. It is widely used to improve the performance on NLP tasks and applications. Another general-purpose knowledge graph that is publicly available is the *ConceptNet* [28], created as part of the Open Mind Common Sense project [11]. This knowledge graph has multilingual properties, where the same concept in two different languages share a common semantic space, which is informed by all other languages. Knowledge graphs have the disadvantage of not being as easy to use as word embeddings; however, it is possible to incorporate this information into word embeddings [26, 35, 27], in order to improve the semantic relations between connected concepts. Speer et al. [28] make use of a set of pre-trained word embeddings, *ConceptNet Numberbatch*, that have been fine-tuned to encompass the relations present in *ConceptNet*, benefiting from the fact that they include semi-structured common sense knowledge. It was built on a combination of data from *ConceptNet*, *Word2Vec* vectors, *GloVe* vectors and *OpenSubtitles*, through a technique called *retrofitting*, to inject the knowledge into the vectors. Another interesting aspect of *ConceptNet Numberbatch* is that the multilingual properties from *ConceptNet* are kept in the embeddings, making it a very interesting resource for multilingual applications.

Retrofitting is a technique firstly introduced by Faruqui el al. [10] and, as previously mentioned, aims at incorporating the data present in semantic lexicons such as *WordNet* or *ConceptNet* into a previously defined word vector space, such as *Word2Vec* or *GloVe*. Hence, this refines the vector space to account for relational information, meaning that words which are lexically linked together should have similar vector representations. Retrofitting works by applying a linear vector transformation to the vectors that closes the gap between related words and increases the distance between lexically unrelated words. The transformation leads to a loss function Ψ that should be minimized, represented in Equation 1, where $\hat{q}_i$ is the initial vector, q_i the retrofitted vector and q_j its neighbors in the ontology. The parameters α and β are hyperparameters that control the relative strength of each parcel in the equation.

$$\Psi = \sum_{i=1}^{n} \left[\alpha_i \left\| q_i - \hat{q}_i \right\|^2 + \sum_{(i,j) \in E} \beta_{ij} \left\| q_i - q_j \right\|^2 \right] \tag{1}$$

In order to minimize the loss represented in Equation 1, it must be differentiated, resulting in Equation 2, which corresponds to the linear transformation applied to the vectors.

$$q_i = \frac{\sum_{j:(i,j)\in E} \beta_{ij} q_j + \alpha_i \hat{q}_i}{\sum_{j:(i,j)\in E} \beta_{ij} + \alpha_i} \tag{2}$$

By carefully analyzing the equation, one can conclude that it corresponds to the weighted average of the initial vector embedding and the vectors representing the concepts that are linked to it, controlled by parameters α and β, where the former attributes increased relevance to the initial vector and the latter controls the importance of the linked concepts.

Regarding food specific embeddings, Food2Vec[1] is a set of pre-trained embeddings that were generated using a corpus of recipe instructions. The goal was to create a recipe recommendation system that joined ingredients in order to create new recipes based on the embeddings of those ingredients. Although the embeddings are specialized for food, they do not capture the semantic relationships between hypernyms and hyponyms among ingredients. Instead, these embeddings encoded the relations that several ingredients have when used together. As an example, according to this methodology, *parmesan* is closer to *pasta* than to *cheddar*, even though *parmesan* and *cheddar* are two types of cheese. This is due to the fact that *parmesan* and *pasta* are used together many times in recipes, whereas it is rather rare to mix *parmesan* and *cheddar* in the same recipe. This approach would not be appropriate

[1] `https://jaan.io/food2vec-augmented-cooking-machine-intelligence/`

for ingredient classification, although it is an interesting approach to new recipes. A similar approach is followed by Tansey et al. [29], which encodes complete diets into a vector space using a combination of Word2Vec embeddings and nutritional information. Moreover, the work of Sauer et al. [24] tries to emulate different ingredient flavors in a vector space. Also, the work of Popovski [21] tries to generate a set of embeddings for a food ontology; however, the authors were not able to really grasp and capture the semantic relations between the concepts, since they only used information from the knowledge graph itself. That being said, there is a lack of NLP tools for food-related applications that are actually able to incorporate and represent the semantic relations between the different concepts and items.

3 Methodology

This work is part of the implementation of a conversational agent into a nutritional recommender system [22, 23], part of the *LiFANA* project [2]: a smart meal planner that takes into account personal information and preferences to create meal plans that are tailored to the user's nutritional needs. The recommendation engine relies on a recipe database that was created by nutritionists based on the McCance and Widdowson's The Composition of Foods[2] integrated dataset [25]. This database contains the nutritional composition of different food and its corresponding classification using *LanguaL* [14][3] descriptors. *LanguaL* is a multilingual thesaurus that allows describing food from different facts. Each food is described by a set of descriptors from different facets pertaining to different perspectives to classify food, including for instance its classification, source and presentation method. A numeric coding is used which allows translating each concept in different languages [14]. The recipes used by the system were built on top of this information by mixing ingredients and quantities. Every ingredient present in the database has at least one *LanguaL* descriptor as a classifier. In this work only facets A (food groups), B (origin) and C (part of animal or plant where the ingredient comes from) are considered, since they are regarded as the ones that effectively describe the ingredient when considering food preferences:

- Facet A – The ingredients are classified according to a food group to which they belong. Facet A gathers several international standards for food grouping. For the purpose of this work, the classification from the European Food Groups will be considered, since it was regarded as the one with the most granularity.
- Facet B – Addresses the food source and has several hierarchical levels. For this work, only the last and more specific level is considered. As examples, *milk*'s food source is *cow* and *raisin*'s food source is *grape*. This facet is particularly important to aggregate foods that correspond to their food source, such as fruits and vegetables or types of fish.
- Facet C – Categorizes the part of the animal or plant from which the ingredient is extracted. To illustrate, the descriptor for *cheese* under this classification is *milk*. This facet presents the least connection to current language terms, although it can be important to discern from similar ingredients semantically.

  An example ingredient classification is illustrated below:
- *Chicken curry, chilled/frozen, reheated*
  LanguaL Descriptors:
  - **A0715** 25 Poultry and Poultry Products (EFG)
  - **B1457** Chicken
  - **C0268** Skeletal Meat Part, Without Bone, Without Skin

[2] https://www.gov.uk/government/publications/composition-of-foods-integrated-dataset-cofid
[3] https://www.langual.org/Default.asp

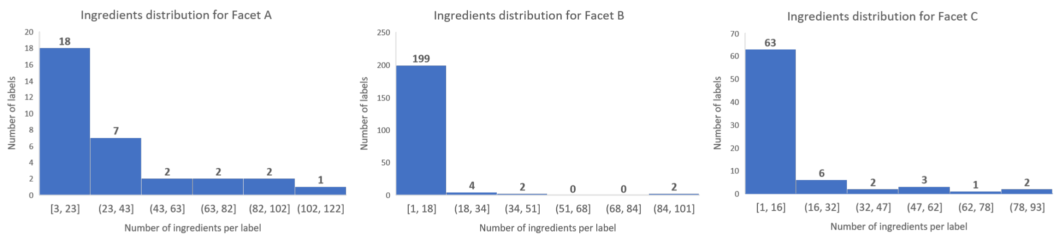


Figure 1 Class distribution histograms for the three facets.

Table 1 Number of class labels per facet and median number of ingredients per label.

Facet	Number of labels	Median of ingredients per label
A	32	14.5
B	207	2
C	77	4

The database of the nutritional recommender system contains 4970 ingredients, from which 902 are labelled according to all three considered facets. Ingredients distribution according to the three facets is depicted in Figure 1. This figure evinces an imbalance in class distribution. Facet A is the one with a less unbalanced number of ingredients, while Facet B presents a large number of labels with less than 10 ingredients, similar to what happens in Facet C. Facets B and C have a few major classes that engulf a lot of ingredients. Furthermore, the total number of labels per facet and the median of the number of ingredients for the labels of each facet is shown in Table 1.

In order to create proper embeddings, two paths could be taken: provide annotated data to the model, such as tuples of a query entity and its database matches; or learn unsupervised embeddings from the relationships between the entities. Still and all, given the available data, which is scarce and not well structured, none of the aforementioned paths is an ideal option for good performance. As a matter of fact, some of the groups of labels have only one ingredient. This will not generalize well for new data, requiring an alternative method. As mentioned in Section 1, retrofitting was used to create a set of pre-trained embeddings.

People often use hypernyms to express their preferences towards broad groups of ingredients, instead of referring to individual ones. The combination of synsets and lexicon databases, such as *WordNet* or *ConceptNet* [28], with the information provided by *LanguaL* facets on each ingredient creates an interesting set of features to classify ingredients. Following the work of Wu et al. [33], the approach designed for this task is based on using or creating embeddings for ingredient classification. The logic behind this method is that any entity can be embedded by a neural embedding model by learning feature representations for relationships among collections of those entities. The vector space is the same for all entities, which enables the model to rank entities, documents, or objects according to the similarity measure to a given query entity.

Bearing this in mind, each group of *LanguaL* facets can be regarded as a set of labels to classify each ingredient. The hypothesis we seek to explore in this work is that a correct classification of the ingredient in each of these three groups may enhance the retrieval of possible candidates from the ingredient database.

4 Pre-trained Embeddings

On a first stage, general pre-trained embeddings, such as *Word2Vec* and *GloVe*, were used to map every ingredient and *LanguaL* facets into a common vector space and hence, enabling us to classify each ingredient in the database as one of the *LanguaL* facets, for each one of the groups. This means that each ingredient was classified under three different groups of labels, one for each facet group.

Each one of the ingredient embedding, as well as each facet name embedding, was created by averaging the embeddings of each word that compose them, excluding stop-words. For that, several steps of pre-processing are required:

1. Tokenization – In order to obtain each individual token, spaCy tokenizer was used
2. Stop-words removal – Each extracted token is compared with a dictionary of stop-words so that they are removed to prevent retrofitting with them
3. Word normalization – Words are normalized in order to remove plural inflections which is particularly common in this dataset
4. Dictionary matching – The last step concerns matching the tokens with the words in the embeddings lexicon to extract the embeddings. However, there are some words that are not present in the lexicon or are not in the correct format. To solve this, the following steps are performed:
   a. Try to match bigrams with the lexicon; for hyphenated tokens, try matching their transformation into bigrams or unigrams (either by concatenating both tokens or by matching them individually).
   b. Perform fuzzy matching using the *FuzzyWuzzy* Python library, which calculates the Levenshtein distance to all possible words in the lexicon and retrieves the most probable candidate that has a distance of 1, if there is any.
   c. If not, create a zero-valued vector to emulate the embedding.

These steps were designed after exploring the database and finding some patterns in its data. For example, regarding the group *Fish and Seafood* (one of the labels for *LanguaL*'s Facet A group), the resulting embedding would be the average of the vectors for *Fish* and *Seafood*. This process is similar for ingredients, that usually have more than one token per name (see the examples in Section 6). Each ingredient and each class label is represented by only one vector, regardless of the number of tokens that compose its name. Class prediction is based on the Cosine Similarity between the embedding of a given ingredient and the embedding of each *LanguaL* class label, for each one of the three facets considered, following the work of Wu et al. [33]. The most similar class, for each facet, is the prediction made by the model. At the end of the classification process, each ingredient has 3 labels: one for each facet. Classification accuracy was used to evaluate the results obtained.

Word2Vec or *GloVe* are generic embeddings trained on large text corpora, and do not encode information extracted from knowledge graphs. We hypothesize that using embeddings that do, such as ConceptNet Numberbatch [28], is a sensible approach to understand whether the semantic information present in knowledge graphs can be leveraged to better enrich the embeddings in this particular context of food terms.

Table 2 summarizes the results obtained for two sets of pre-trained embeddings: GloVe (which incorporates no knowledge graph information) and ConceptNet Numberbatch (retrofitted with knowledge graph information).

As it is possible to note, the results obtained by using Numberbatch embeddings are higher than the ones obtained using GloVe. This shows that retrofitting does incorporate some semantic information into the embeddings, that leads to better semantic relationships between the different ingredients' embeddings.

5 Embeddings Refinement with Retrofitting

Being *LanguaL* an ontology which incorporates lexical information about food in a knowledge graph style, these established relations may be used to retrofit the pre-trained Numberbatch's embeddings so that they become more aware of food semantics. Thus, as a second step, the Numberbatch embeddings were retrofitted with the information that is available in *LanguaL*.

There are three possible ways to perform retrofitting with the available data:

A. Retrofitting class embeddings with the vectors retrieved for each of the ingredients that have that class as a descriptor in the database.
B. Retrofitting ingredient embeddings using the vectors of the classes that they have as descriptors in the database.
C. Combining the two previous approaches and retrofit both the ingredients and the classes.

Each procedure presents advantages and disadvantages. Procedure **A** will not change the embeddings associated with each ingredient. Instead, it encodes each *LanguaL* class according to the ingredients that it contains. This may pose as an advantage for real world applications when dealing with ingredients that are not part of the database. In these cases, provided that there are similar ingredients in the database, the model would still be able to classify the query. On the other hand, procedure **B** changes the vectors that represent the ingredients in order to match the *LanguaL* class label embedding. It is expected that this method will converge easily since each ingredient will only be retrofitted using, at the most, 3 concepts, which does not happen in the former method. Finally, procedure **C** tries to change both elements towards a converging representation. This will adapt both data types to each other which may be harmful when handling new information. This simultaneous change may cause the model to not generalise well for new ingredients as well as for new class labels, in case they are added to the database. Also, convergence may not be achieved since some of the concepts (ingredients and labels) are related in different ways. From iteration to iteration, the embeddings are being changed according to different embeddings (since both the ingredients and the class labels embeddings change). The selected approach should consider the performance obtained, after retrofitting, for all three facet groups at the same time, since there will only be one embedding representation per ingredient. Retrofitting each facet group individually and sequentially may harm the scores of the previously retrofitted facet groups. Even though one method may provide better results than another for an individual facet, the joint performance in the three facets should be considered. The results were obtained using k-fold cross validation with 6 randomly selected folds, (see Table 3). The folds could not be stratified since there are three simultaneous classification problems being addressed. Also, this means that some folds might not have data for every class in each of the three facets. The retrofitting session lasted for 20 iterations, after each of which the results were validated using the cosine similarity criterion, to consider updating the β

Table 2 Accuracy results for ingredient classification according to *LanguaL* facets, using different pre-trained embeddings models.

Model	Accuracy per facet		
	A	B	C
GloVe 400k vocabulary	0.277	0.324	0.087
ConceptNet Numberbatch	**0.417**	**0.401**	**0.089**

Table 3 Accuracy results for ingredient classification according to *LanguaL* facets, using three different retrofitting procedures.

Procedure	Accuracy		
	Facet A	Facet B	Facet C
Baseline results (no retrofitting)	0.412 ± 0.044	0.447 ± 0.027	0.108 ± 0.023
Approach A	$\mathbf{0.648 \pm 0.029}$	$\mathbf{0.505 \pm 0.050}$	$\mathbf{0.402 \pm 0.046}$
Approach B	0.412 ± 0.044	0.447 ± 0.027	0.108 ± 0.023
Approach C	0.504 ± 0.054	0.417 ± 0.050	0.242 ± 0.035

parameter (see Equation 2). If the results increased overall (throughout the 3 facets), β is maintained for the next iteration; otherwise, its value is increased by a factor of 20%. The initial β value is set to 1, the same as α; however, the latter is fixed for the whole session. At the starting point, both the vector to be retrofitted and the definitions have the same weight, which allows not to lose intrinsic information about the words, since the concept to be retrofitted should still retain some semantic meaning in order to prevent overfitting and generalize better to new data.

The results were obtained by evaluating the accuracy using 6-fold cross validation, since it was the maximum number that allowed to have all classes represented in each fold. By looking at Table 3 it is clear that the procedure that produces the best overall results is procedure A, where the classes were retrofitted incorporating information about the ingredients. This proved the hypothesis that procedure A deals better with unseen data than the other procedures, by not altering the values of the ingredient embeddings. During the training of procedure B it was not possible to make the model converge as the global accuracy was not stabilizing in a value, instead it was increasing and decreasing around the base value. This makes sense when thinking about the testing process. The ingredient embeddings of the training set are being altered according to the *LanguaL* information; however, the ones in the validation set have not suffered this alteration. This means that the classification score will be mostly the same in this case. Regarding procedure C, it is possible to see an improvement in both facets A and C, although the accuracy in facet B decreases. Bearing in mind these results and the perceived good handling of new data, procedure A was selected as the method to create the new set of embeddings that will be used in ingredient retrieval.

6 Token TF-IDF Weighting

Despite the fact that there is a clear improvement in classification accuracy, due to the naming format of the ingredients, the embeddings may be considering information that is not relevant for classification. The following list illustrates some examples of ingredient names present in the ingredients database:

- Pineapple, canned in juice
- Eggs, chicken, whole, raw
- Onions, raw
- Tuna, canned in brine, drained
- Peppers, capsicum, green, boiled in salted water

In most cases, ingredient names include extra information that may not be relevant for classification, such as the cooking method or the way of preservation. Moreover, the order of the words in the ingredients´ names does not always follow the same logic because the ingredients derive from several sources. As a consequence, it was not possible to create rules for name processing before retrofitting. An example of such a rule would be to remove every word after the first comma; however, as is noticeable in the examples above, some important characteristics regarding ingredients are present after the first or even second comma. This pre-processing would have to be hand-made, which would be impractical. As a way to deal with this problem, a weighting mechanism based on the Term Frequency - Inverse Document Frequency (TF-IDF) weighting was applied to the tokens during ingredient and class label embedding creation. This way, words that are not important to distinguish classes will have reduced importance in the embeddings for retrofitting. TF-IDF weights were calculated in three different ways, depending on what was considered a document:

1. Concatenating ingredient names for a given class label
   In order to illustrate this case, when retrofitting the class label *Fish and Seafood* a document would comprise all ingredient names that belong to that class. This would boost the Term-Frequency part since there are usually several ingredients with similar names, varying only the cooking method, for example. Also, this would punish words that appear in different concepts, such as the cooking or preservation methods that are common to different types.
2. Considering each individual ingredient/class name
   In this case, Term-Frequency will not benefit, although Inverse Document Frequency will punish even more the tokens that appear in many ingredient or class names
3. Hybrid approach: the Term Frequency is calculated through procedure 1 while the Inverse Document Frequency is calculated using procedure 2
   With this hybrid approach the goal is to further punish words that appear in many classes while boosting words that belong to only one class.

Embeddings were retrofitted once again using method A with each of the TD-IDF approaches. We present and analyze the results obtained for the retrofitting procedure A, which gave the best results for the ingredient classification task under the three facet groups, as shown in Table 3, with an extra step of token weighting before constructing the ingredient or class label embedding. The results are presented in Table 4. Once again, they were taken using 6-fold cross validation. We can observe that TF-IDF weighting improves the results. Every experimented method increased performance when comparing to retrofitting without weighting. However, it is clear that TF-IDF 3, the hybrid method, presents the best overall results, showing the largest improvements in all three groups of labels. The differences between TF-IDF 2 and 3 have for all three facets have statistical significance. However, between TF-IDF 1 and 3 there is no statistical significance. TF-IDF 1 presented very similar results for facet B and facet C, although the error values are higher and the accuracy for facet A is lower. TF-IDF 2 presented the worst results of the three. Even though the differences between TF-IDF 1 and TF-IDF 3 have not statistical significance, the hybrid weighting method was slightly better at dealing with this kind of data due to selectively punishing the terms according to their frequency in different groups.

The evident improvement to using generic pre-trained embeddings shows that we were successful in incorporating food semantic information available in *LanguaL* into word vectors. The next section explores the algorithm designed to classify and retrieve ingredients based on a query entity, which takes advantage of a set of pre-trained ConceptNet Numberbatch embeddings retrofitted with *LanguaL* semantic information, through approach A and using the hybrid TF-IDF weighting (TF-IDF 3).

Table 4 Accuracy results for ingredient classification according to *LanguaL* facets, using approach A for retrofitting with different methods of TF-IDF weighting.

TF-IDF approach	Accuracy		
	Facet A	Facet B	Facet C
Baseline results (no TF-IDF weighting)	0.6475 ± 0.029	0.505 ± 0.050	0.402 ± 0.046
TF-IDF 1	0.690 ± 0.031	0.595 ± 0.064	0.451 ± 0.031
TF-IDF 2	0.682 ± 0.041	0.563 ± 0.061	0.411 ± 0.047
TF-IDF 3 (hybrid)	**0.692 ± 0.026**	**0.595 ± 0.048**	**0.451 ± 0.021**

Algorithm 1 Matching and Extracting Algorithm.

```
1  Input: Sentence, th1, th2, th3, th4
   Result: List of database ingredients that match the query entity
2  Ingred = getNamedEntity(Sentence)
3  IngredEmbedding = getEmbedding(Ingred)
4  [classA, classB, classC] = classifyIngred(IngredEmbedding)
5  EmbSimList, FuzMatchList = extractPossibleMatches([classA, classB, classC], th1,
    th2)
6  CandidateList = EmbSimList $\bigcap$ FuzMatchList
7  if length(CandidateList) > 1 then
8      probableGroupMatch = calculateGroupMatchEmb(IngredEmb, classA, classB,
        classC, th3)
9      groupMatch = calculateGroupMatchLev(Ingred, probableGroupMatch, th4)
10     if length(groupMatch) > 0 then
11         Return extractFromDB(groupMatch)
12     else
13         Return CandidateList
14 else if length(CandidateList) == 1 then
15     Return CandidateList
16 else
17     Return []
```

7 Food Matching

The purpose of creating word embeddings that capture the semantic relations present in the *LanguaL* ontology was to retrieve, from the recommender system's ingredient database, the relevant ingredients, given a query entity extracted from user input. A perfect retrieval process would gather all ingredients that correspond to the Named Entity query based on the classification of the words (ingredients) that compose it. The entity may point to a group of ingredients, to a specific ingredient, or even to a group of ingredients that do not match *LanguaL* labels exactly. Bearing this in mind, a matching and extraction algorithm was developed. This algorithm leverages the food information that was incorporated in the embeddings and is detailed as Algorithm 1.

The first step is the Named Entity Recognition (line 2), in order to identify the ingredient that is present in the user's query. This entity, which represents a name of an ingredient, is then preprocessed, as described in Section 4, and encoded into an embedding (line 3). The next step is the classification of the query according to the three facets (line 4). Using the predicted class labels, the algorithm extracts all ingredients that match at least 2 of the 3 classes from the database, resulting in a first list of possible candidates. Moreover, this list is then filtered using two criteria: embeddings cosine similarity and fuzzy matching to create two lists of ingredients that are strong candidates (line 5). Both these filters require predefined thresholds, th1 and th2, respectively. These previously mentioned lists are intersected in order to remove options from the fuzzy match search since it produces a large list with some unrelated ingredients. The result is the final list of candidates that will be the input to a series of conditions that will define the final result (line 6).

In case the Candidates List has only one ingredient, this ingredient is regarded as the match to the query and is returned by the system (lines 14 and 15). If the Candidates List has no elements, an empty array is returned, which means that there are no matches in the database for the ingredient query (lines 16 and 17). On the other hand, if the Candidates List has several elements, there is a strong possibility that the query is referring to a group of ingredients, rather than to a single ingredient. The query is then compared to the predicted class labels in a sequential process that uses cosine similarity and Levenshtein distance, requiring two threshold values: th3 and th4 respectively). The result of this last process (lines 8 and 9) is a list of *LanguaL* labels that may match the query. If this list is not empty, then the algorithm extracts all ingredients that are labeled accordingly (giving priority to facet A, then B and lastly C) (line 11). Otherwise, the Candidates List is returned. This means that the query matches a group of ingredients that is not a specific *LanguaL* group.

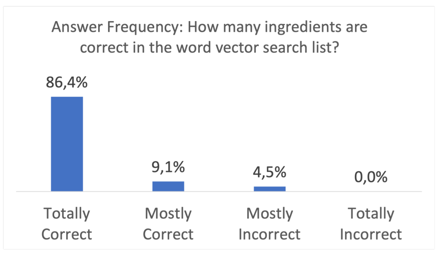
The thresholds referred in the algorithm were defined through observation, in order to maximize accuracy. An increase in the value of the threshold would represent an increase in precision with a consequent decrease in recall, since it causes a decrease in the number of ingredients retrieved. It makes the model more certain about the ingredients its extracting with the drawback of maybe missing some correct ones. This method was validated by user testing, as explained in Section 7.1, since there is a lack of an annotated dataset that could serve as validation.

7.1 User Validation

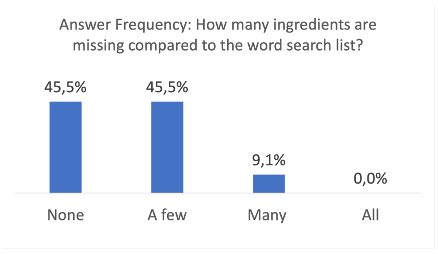
Due to the lack of a proper dataset, the matching algorithm was qualitatively evaluated by volunteers. Each volunteer must suggest one ingredient that has not yet been selected by others. Both the results obtained through the ingredient retrieval algorithm (based on embeddings) and the ones obtained by using a fuzzy word matching algorithm are shown to the volunteer, who has to answer three questions. This last search method is regarded as baseline method and was the only implemented in the recommender system. It compares the strings of the ingredients character by character.

1. Whether or not the ingredients retrieved by the embeddings-based algorithm are correct, on a 4-valued scale, from “totally incorrect” to “totally correct”.
2. Whether or not there is any ingredient obtained from word matching search that should also be in the embeddings results list, on a 4-valued scale, from “none” to “all”. It is worth noticing that the word matching search results usually contain ingredients whose name is similar to the one in the query, even though they do not match (e.g., *cheesecake* is a result of searching for *cheese*).
3. What is the preferred option for ingredient retrieval.

This way it is possible to evaluate the developed algorithm from a user standpoint, using some approximate recall and precision metrics. The test was performed by 22 volunteers (one ingredient each). Figure 2a shows the answer distribution for the question that addressed the



(a) Response frequency for food matching precision assessment.



(b) Response frequency for food matching recall assessment.

Figure 2 Qualitative results from the user ingredient retrieval evaluation.

correctness of the shown results by the embeddings-based search algorithm. These results can be mapped to the precision of the system, which measures how many of the positive answers are in fact true. Results show that the developed algorithm has a high level of precision. The large majority of the query outputs are totally correct, meaning that the ingredients the system shows are in fact related to the query term.

Recall is also another important metric that should be taken into account. Figure 2b shows the answers regarding the comparison made between the word-based search and the developed algorithm. The goal was to identify items that were correctly present in the former and missing in the latter. This does not calculate the true recall of the model, which would require a list of all correct items per query. Nonetheless, it is a useful comparison to detect missing items. The results show that the large majority of the answers were positive, meaning that the possible recall is also high. However, it is possible to affirm that the model shows increased precision when compared to recall. According to the answers that judged the preferred algorithm, the embeddings based algorithm explained in this paper was preferred by **77.3%** of the users. Even though the recall is lower than precision and the algorithm does not always gather all ingredient samples from the dataset, users prefer to have access to correct ingredients.

8 Conclusions

This work described the incorporation of semantic information from a food-related ontology into word embeddings, hence creating a set of embeddings that really capture the relations between food terms. Pre-trained embeddings were shown to poorly encode the different linkages that exist between food terms, creating the necessity of more semantic-aware embeddings. These relations are hard to capture with text data. Retrofitting has shown to be a valuable technique that enabled the enrichment of general knowledge embeddings in terms of food relations, largely increasing class similarity between the descriptor labels from *LanguaL* and the ingredients names. Also, from the three methods that were tried for retrofitting, regarding what embeddings to change (the class labels or the ingredients), the one that provided the best results was to alter the embeddings from the labels, based on the embeddings from the ingredients. This means that even if new ingredients are to be classified, this method should still be able to classify it correctly, leading to better results in the cross-validation testing. Moreover, TF-IDF weighting in the embedding creation proved to improve the results by giving different importance to the tokens that compose each ingredient and class name. This way, only words that are really specific and distinctive from each name are used to perform the retrofitting of the embeddings. TF-IDF was calculated in three different ways, from which the one that gave the best results was a hybrid approach where the TF and IDF parts were calculated using different concepts of "Document". The implementation of this weighting mechanism allowed for an uptick in the class prediction accuracy. Further validation of this method and the resulting embeddings may be made to properly fine tune the embeddings. A validation data set may be created to analyze the methods and establish benchmark scores for several parameters such as measuring ingredient similarity through embeddings and comparing it to real life similarity. Even though the evaluation presented above showed very promising results, an exact and quantitative method should also be applied in order to further validate and reinforce these conclusions.

This work resulted in a set of pre-trained embeddings that already incorporate food knowledge and can be used for several applications besides database extraction and ingredient classification. These embeddings may pose as a useful tool for nutrition recommender systems or health companions in functionalities such as ingredient substitution or recipe creation by leveraging the ingredient relations and similarities. Another example of a possible application is the classification of recipes based on the ingredients that compose them, which then can be used to generate meal suggestions fostered by the similarity between user preferred meals and new ones. The applications that can be powered by these embeddings should also be properly validated by the creation of benchmark tests and scores. Moreover, this work evinces that the application of retrofitting as a way of enriching embeddings can be applied to virtually any context that requires grasping semantic relations, as long as there is a knowledge graph or similar structure that encodes these relations to support it.

References

1 Alaa A. Abd-alrazaq, Mohannad Alajlani, Ali Abdallah Alalwan, Bridgette M. Bewick, Peter Gardner, and Mowafa Househ. An overview of the features of chatbots in mental health: A scoping review, December 2019.

2 Andreas Arens-Volland, Benjamin Gateau, and Yannick Naudet. Semantic Modeling for Personalized Dietary Recommendation. *Proceedings - 13th International Workshop on Semantic and Social Media Adaptation and Personalization, SMAP 2018*, pages 93–98, 2018. `doi:10.1109/SMAP.2018.8501864`.

3 Ram G. Athreya, Axel Cyrille Ngonga Ngomo, and Ricardo Usbeck. Enhancing Community Interactions with Data-Driven Chatbots - The DBpedia Chatbot. In *The Web Conference 2018 - Companion of the World Wide Web Conference, WWW 2018*, pages 143–146, New York, New York, USA, April 2018. Association for Computing Machinery, Inc. `doi:10.1145/3184558.3186964`.

4 Timothy W. Bickmore, Daniel Schulman, and Candace L. Sidner. A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology. *Journal of Biomedical Informatics*, 44(2):183–197, April 2011. `doi:10.1016/j.jbi.2010.12.006`.

5 Kyungyong Chung and Roy C. Park. Chatbot-based heathcare service with a knowledge base for cloud computing. *Cluster Computing*, 22(1):1925–1937, January 2019. `doi:10.1007/s10586-018-2334-5`.

6 Leigh Clark, Nadia Pantidi, Orla Cooney, Philip Doyle, Diego Garaialde, Justin Edwards, Brendan Spillane, Emer Gilmartin, Christine Murad, Cosmin Munteanu, Vincent Wade, and Benjamin R. Cowan. What makes a good conversation? Challenges in designing truly conversational agents. In *Conference on Human Factors in Computing Systems - Proceedings*, pages 1–12, New York, New York, USA, May 2019. Association for Computing Machinery. `doi:10.1145/3290605.3300705`.

7 Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 1, pages 4171–4186, 2019. URL: `https://github.com/tensorflow/tensor2tensor`.

8 Vanesa Espín, María V. Hurtado, and Manuel Noguera. Nutrition for Elder Care: A nutritional semantic recommender system for the elderly. *Expert Systems*, 33(2):201–210, 2016. doi:10.1111/exsy.12143.

9 William J. Evans and Deanna Cyr-Campbell. Nutrition, exercise, and healthy aging. *Journal of the American Dietetic Association*, 97(6):632–638, 1997. doi:10.1016/S0002-8223(97)00160-0.

10 Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. In *NAACL HLT 2015 - 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, pages 1606–1615, 2015. doi:10.3115/v1/n15-1184.

11 Catherine Havasi, Robert Speer, Kenneth Arnold, Henry Lieberman, Jason Alonso, and Jesse Moeller. Open mind common sense: Crowd-sourcing for common sense. In *AAAI Workshop - Technical Report*, volume WS-10-02, page 51, 2010. URL: www.aaai.org.

12 Shafquat Hussain, Omid Ameri Sianaki, and Nedal Ababneh. A Survey on Conversational Agents/Chatbots Classification and Design Techniques. In *Advances in Intelligent Systems and Computing*, volume 927, pages 946–956. Springer Verlag, 2019. doi:10.1007/978-3-030-15035-8{_}93.

13 H. N. Io and C. B. Lee. Chatbots and conversational agents: A bibliometric analysis. In *IEEE International Conference on Industrial Engineering and Engineering Management*, volume 2017-December, pages 215–219. IEEE Computer Society, February 2018. doi:10.1109/IEEM.2017.8289883.

14 J. D. Ireland and A. Møller. Langual food description: A learning process. *European Journal of Clinical Nutrition*, 64:S44–S48, 2010. doi:10.1038/ejcn.2010.209.

15 Veton Kepuska and Gamal Bohouta. Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home). In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference, CCWC 2018*, volume 2018-January, pages 99–103. Institute of Electrical and Electronics Engineers Inc., February 2018. doi:10.1109/CCWC.2018.8301638.

16 Stefanie Mika. Challenges for nutrition recommender systems. *CEUR Workshop Proceedings*, 786:25–33, 2011.

17 Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. International Conference on Learning Representations, ICLR, 2013.

18 George. Miller and Princeton University. Cognitive Science Laboratory. *WordNet*. MIT Press, 1998.

19 Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word representation. In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1532–1543, 2014. doi:10.3115/v1/d14-1162.

20 Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 1, pages 2227–2237. Association for Computational Linguistics (ACL), February 2018. doi:10.18653/v1/n18-1202.

21 Gorjan Popovski, Bibek Paudel, Tome Eftimov, and Barbara Korousic Seljak. Exploring a standardized language for describing foods using embedding techniques. In *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, pages 5172–5176. Institute of Electrical and Electronics Engineers Inc., December 2019. doi:10.1109/BigData47090.2019.9005970.

22 David Ribeiro, João Machado, Jorge Ribeiro, Maria João M. Vasconcelos, Elsa F. Vieira, and Ana Correia De Barros. SousChef: Mobile meal recommender system for older adults. In *ICT4AWE 2017 - Proceedings of the 3rd International Conference on Information and Communication Technologies for Ageing Well and e-Health*, pages 36–45. SciTePress, 2017. `doi:10.5220/0006281900360045`.

23 David Ribeiro, Jorge Ribeiro, Maria João M. Vasconcelos, Elsa F. Vieira, and Ana Correia de Barros. SousChef: Improved meal recommender system for Portuguese older adults. *Communications in Computer and Information Science*, 869:107–126, 2018.

24 Christopher R Sauer and Alex Haigh. Cooking up Food Embeddings Understanding Flavors in the Recipe-Ingredient Graph, 2017.

25 Nuno Silva, David Ribeiro, and Liliana Ferreira. Information extraction from unstructured recipe data. *ACM International Conference Proceeding Series*, Part F1482:165–168, 2019. `doi:10.1145/3323933.3324084`.

26 Vivian S Silva, Andre Freitas, and Siegfried Handschuh. Building a knowledge graph from natural language definitions for interpretable text entailment recognition, 2018. URL: `http://brat.nlplab.org/`.

27 Vivian S Silva, Siegfried Handschuh, and Andre Freitas. Categorization of semantic roles for dictionary definitions, 2018. URL: `https://www.aclweb.org/anthology/W16-5323`.

28 Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press, 2017.

29 Wesley Tansey, Edward W. Lowe, and James G. Scott. Diet2Vec: Multi-scale analysis of massive dietary data. *arXiv*, December 2016. `arXiv:1612.00388`.

30 Christoph Trattner and David Elsweiler. Food Recommender Systems Important Contributions, Challenges and Future Research Directions, November 2017. `arXiv:1711.02760`.

31 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

32 Anu Venkatesh, Chandra Khatri, Ashwin Ram, Fenfei Guo, Raefer Gabriel, Ashish Nagar, Rohit Prasad, Ming Cheng, Behnam Hedayatnia, Angeliki Metallinou, Rahul Goel, Shaohua Yang, and Anirudh Raju. On Evaluating and Comparing Conversational Agents. In *Conversational AI Workshop at the 31st Conference on Neural Information Processing Systems*, pages 1–10, 2017. URL: `http://alborz-geramifard.com/workshops/nips17-Conversational-AI/Papers/17nipsw-cai-evaluating_conversational.pdf`.

33 Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. StarSpace: Embed all the things! In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 5569–5577. AAAI press, September 2018.

34 Anbang Xu, Zhe Liu, Yufan Guo, Vibha Sinha, and Rama Akkiraju. A new chatbot for customer service on social media. In *Conference on Human Factors in Computing Systems - Proceedings*, volume 2017-May, pages 3506–3510. Association for Computing Machinery, May 2017. `doi:10.1145/3025453.3025496`.

35 Wen Zhou, Haoshen Hong, Zihao Zhou, and Stanford Scpd. Derive Word Embeddings From Knowledge Graph, 2019.