WordGraph2Vec: Combining domain knowledge with text embeddings

Paul Keuren[1], Marc Ponsen[1], and Ayoub Bagheri[2]

[1] Statistics Netherlands
[2] Utrecht University

Abstract. Many domains lack the data required to train a neural network to solve their issues. For the labour market, the matching of Dutch free-text skills to a predefined taxonomy is such a case where a lack of examples exist. In this paper, we propose a novel algorithm, WordGraph2Vec (WG2Vec) which combines, word embeddings, language models and the domain knowledge they bear. As a first step, WG2Vec performs a domain-specific cleaning of a sentence before dissecting the text. The dissection technique relies on dependency tree parsing. From the dependency graph, a set of domain-specific relations are extracted and converted to pairs of interest (so-called wordgraphs). Wordgraphs should represent the important phrases in the (larger) text. As a next step, the semantics for wordgraphs are obtained using word embeddings models. WG2Vec uses the resulting embeddings combined with literal one-on-one matches to compare wordgraphs from skill descriptions with the ones from taxonomy texts. Analyzing texts both in a grammatical and semantic level, the WG2Vec algorithm can find matches even if completely different wordings are used. In our experiments, the proposed algorithm was tested against the state-of-the-art pre-trained models (paragraph2vec, universal sentence encoding, sentenceBERT). For the given task of matching skill descriptions from job vacancies and taxonomy texts, the WG2Vec algorithm outperforms the other methods. An important side-note of the proposed algorithm is that due to its setup, not every sentence will be converted to a set of wordgraphs which results in better interpretability. This beneficiary filtering combined with embedded domain knowledge could explain why the algorithm outperforms general-purpose algorithms. Additionally, unlike other models, not a lot of data are required to refine the proposed method. This makes the algorithm interesting for domains where large training sets are not available.

Keywords: Language models, word embedding, sentence embedding, natural language processing, wordgraph

1 Introduction

It is estimated that the bulk of today's data is unstructured. This so-called Big Data contains huge amounts of information relevant to statistical institutes such as Statistics Netherlands (CBS). A large portion of the data available to CBS is

categorised and structured by the individual filling out a questionnaire. The issue with questionnaires is that they capture a fixed set of variables and require a lot of testing and development before even sending it to the recipients. This process is slow and labour-intensive. A solution would be to be able to use unstructured data. One such form of interesting data is in text.

Text analysis is specifically interesting to CBS given the COVID-19 crisis and its expected impact on the labour market [11]. Due to the expected increase, there is a need for more insightful and fine-grained statistics. This is made possible by working together with Dutch Employee Insurance Agency (UWV). UWV has amassed vacancy texts from multiple sources. The only structure in this data is the differentiation between a title and a body. Oftentimes the title describes the job title and the body contains a mixture of company description, skills, benefits, compensation and so forth. This mixture of information in the body makes it non-trivial for an algorithm to distinguish between candidate requirements and company descriptions. Being able to use this data would help CBS in improving statistics.

Since text data do not have a predefined structure and might be noisy, deriving information from texts is challenging. Traditional statistical techniques used at statistical institutes, are therefore ill-suited to analyse such data. In recent years, great strides forward have been made in dealing with this type of data, most notably in the field of data mining, natural language processing (NLP) and text analytics. To get from text to a statistic, some standardisation has to take place. This standardisation can be done by trying to fit the text into a knowledge system like a taxonomy. A taxonomy is a non-cyclic directed graph, the nodes can have different types and contain different data, yet the relations are not labelled with type information. By moving down the taxonomy, following the direction of the relations one gets more fine-grained data. Moving up the taxonomy (against the direction of the relations) one gets more abstract data. For the labour market, UWV has such a taxonomy of skills.

The task of matching a skill from a vacancy to an item in the taxonomy could be regarded as a form of translation. The taxonomy is regarded as a more formal language in comparison to vacancy texts. As such methods which are used for translation [3, 10] could be applied in this case as well. These algorithms are good at finding similar words or sentences given enough data. The problem however lies in the training. As the taxonomy is handcrafted and therefore limited in size, using it to train a custom variety of these machine learning algorithms is not feasible. Also, the task of identifying which parts make a good match between skill texts and taxonomy texts will require a lot of annotations. Given the cost and limited time of experts, this is deemed to be feasible. Therefore we need an algorithm which can leverage the similarity knowledge of the machine learning-based approach and is assisted in finding which parts of a text are relevant to this domain.

In this paper, we propose a novel algorithm WordGraph2Vec (in short WG2Vec) to analyse unstructured text data. The proposed algorithm combines two aspects of NLP: so-called **wordgraphs** and word embeddings. First, wordgraphs

are used to dissect and understand a text on a grammatical level, i.e., what is the role of words and how do they link to each other? As a next step, the semantics for a wordgraph are obtained using word embedding models, such as Word2Vec. Words and phrases will be converted into a vector of numbers, where the semantic meaning is captured in the numerical vector (i.e., the phrases with vectors close to each other hold the same semantic meaning). These two steps are at the core of the proposed WG2Vec algorithm.

We present experimental results that show that WG2Vec outperforms current state-of-the-art machine learning algorithms for our skill-matching use-case. In this use-case, we match skills obtained from vacancy texts with validated skills in an expert system. We compare the matches from either algorithm with expert annotations made by the UWV.

The goal of this particular use-case is to analyse vacancy texts to support the UWV in efficiently maintaining their expert system (a labour market taxonomy of jobs and connected skills). The maintenance of this expert system consists of two tasks: (1) finding synonyms for existing skills in online texts, and (2) finding new skills and jobs currently absent in the taxonomy. This up-to-date system will then create a 'universal language of jobs and skills', which is relevant for both UWV and CBS. For UWV, this allows them to efficiently match supply and demand on the granular level of skills. CBS, on the other hand, can compute (timely) demand for specific skills and (potentially future) jobs. Combining this timely and detailed view of the labour market with the already existing register data at CBS can potentially provide new insights and statistics for a rapidly developing labour market.

In this paper, we first review existing work related to our research. This is followed by our algorithm and how it is used to measure sentence similarity. After that, we show how the hyperparameters and how the algorithm was measured and the outcome of the measurements. We conclude with a general overview and point out possible future research paths.

2 Related Work

In this section, we describe related work on different types of embedding (word and sentence vectorization) followed by a similar domain issue dealing with matching in the labour market.

Word vectorization: In literature, there is a lot of focus on embedding words (Word2Vec [6], Glove [8], BERT [4]). These techniques focus on creating a vector representation which encompasses the meaning of a word. The vector representing a word is oftentimes based on a prediction task. Given a sentence with one masked word the algorithm has to predict the masked word. It can also be trained the other way around by masking neighbouring words and given a single word the algorithm has to predict the masked neighbours. These techniques are unsupervised and, given enough data, can result in usable vectors at a word level. The main shortcoming of these word vectorization methods is that

a word can have a different meaning in a different sentence. As such the sentence in which the word is present is of importance.

Sentence vectorization: To take into account the meaning of a sentence many algorithms build on the word vector algorithms. Examples of these algorithms are Paragraph2vec [5] and Sentence BERT (sBert) [10] and Universal Sentence Encoding (USE) [3]. Each of these methods has its way of reducing the list of word vectors to a sentence vector representing its content. The output for each of the above methods is a vector of a fixed length despite the input having a fluctuation in the number of words. The techniques are often validated by comparing sets of curated sentences from multiple domains [2].

Labour market matching: The domain of interest to our research is the labour market. Specifically the skills and the way they are described from a vacancy. It shares some problems with job titles as they can also be rather noisy. The research done by Yamashita et.al. [12] shows how complex the matching of noisy Job titles is to a standard form. The input to the algorithm already contains the title and tries to find the closest standard notation. Skill matching to a taxonomy or standard form is not that often done in literature and differs from the Job titles.

3 WordGraph2Vec

In this section, we explain the proposed WordGraph2Vec algorithm. It was developed to detect similarities in text and combines components from the techniques discussed in the previous section. The first component is word embedding. It translates words into vectors so that semantic similarities between words can be measured mathematically. The other component is the language model, specifically dependency-tree parsing and part-of-speech (POS) tagging. The taxonomy, available for the use-case of matching skills, is not exploited by our method. This decision was made to make the method more widely applicable. The combination of word-embedding with dependency-tree parsing and POS tagging are used to find relations between words which are of interest to the domain. These combinations between words are called **wordgraphs**.

The WG2Vec algorithm combines the two components, wordgraphs and word embedding, in such a way as to detect similarity in text information by joining a syntactical understanding of sentences with a semantic representation of words.

We will discuss all steps involved in WG2Vec in the following subsections.

3.1 Understanding a Sentence

Sentences can be noisy and complex constructs. They may contain boilerplate text, which may not give very insightful information about the meaning. Furthermore, people might concatenate sentences, for example:

– “You drive big trucks and cars.”

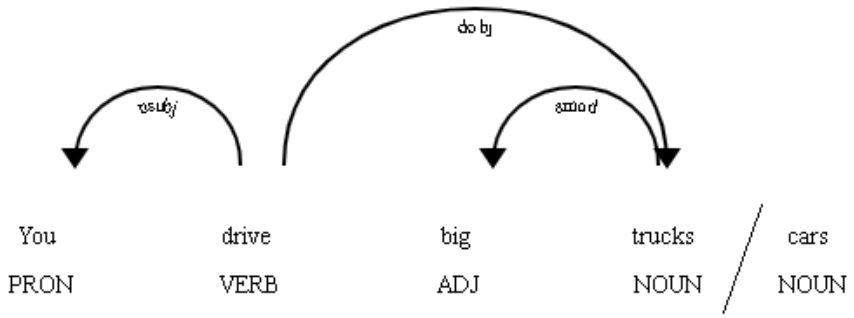

Fig. 1: A language model provides additional details on words, such as their word type (e.g., noun, adjective etc.) and their dependency to other words. WG2Vec uses this information to extract relevant parts from sentences.

The problem with this sentence is the number of combinations that can arise from these sentences. The example shown above has two different pieces of information. Each of these pieces requires a separate analysis to find the best match. For this reason, WG2Vec first pre-processes sentences and splits them into elementary parts. This is done based on a rule-based logic that uses information provided by the language model. The sentence mentioned before could be split into multiple parts:

- "You drive big trucks."
- "You drive big cars."

These elementary pieces of information can be further analysed to better understand their meaning. Techniques such as Paragraph2vec will incorporate boilerplate text in their learned sentence embedding[5]. We hypothesise that better results are obtained once we exclude them. However, pinpointing the relevant parts of a sentence is not an easy task. WG2Vec uses part-of-speech tagging with dependency-tree parsing provided by language models [7] to identify relevant parts in the pre-processed texts.

The method uses this dependency-tree parsing (which is a type of graph, where the words are nodes and their relations are the edges), the resulting combinations of words are named wordgraphs. An additional benefit of this selection proces is the automatic filtering of stopwords. If a word with a specific role in a sentence is not of interest, it does not result in a wordgraph, hence explicit removal of such words is not needed.

Let us give three examples of *wordgraphs* that were tailored to the labour market use-case. The reasoning in how the wordgraphs are constructed shows how expert knowledge is encoded into the system. In this use-case are particularly interested in which skills people have. Therefore, we defined an **Activity** wordgraph. A set of two words is an **Activity** if and only if a VERB refers to a NOUN, return "VERB NOUN". The activities in our example sentence are (see Figure 1):

– "drive (VERB) trucks (NOUN)"
– "drive (VERB) cars (NOUN)"

Within this activity, we may believe that the noun may hold more or less relevance compared to the verb. For that reason, we explicitly also define a **Leading** wordgraph. So in our example sentence, both "trucks" and "cars" are **Leading** wordgraphs.

Finally, we might lose specific information when we only focus on VERB-related information stated above. When the author of a text gave more information on a specific noun, he or she did so for a reason. To incorporate this, we introduced a **Specific** wordgraph.

A set of two words is a **Specific** if and only if an ADJECTIVE refers to a NOUN, return only the "ADJECTIVE NOUN". If a group of ADJECTIVES refer to the same NOUN, each ADJECTIVE gets a separate **Specific** wordgraph. In our example sentence, this yields the following specifics:

– "big (ADJECTIVE) trucks (NOUN)"
– "big (ADJECTIVE) cars (NOUN)"

It is not always the case that these parts are found in sentences. As fallback single nouns and verbs are also used as wordgraphs. The selected wordgraphs are based on this specific use-case, it is possible to concatenate more words into longer wordgraphs. All these parts are taken into account when representing the text in vector space, as will be described in the next Section.

3.2 Representing a Sentence as a Vector

In the next step, we convert the wordgraphs to an embedding. The selected embedding algorithm is Word2Vec [6], although any model could be used to serve as the basis for the algorithm. For each word in the wordgraph, a vector is constructed. Wordgraphs containing more words would automatically result in more vectors for the wordgraph. Let us illustrate using a wordgraph comprising two words w_1 and w_2, each with a length of m.

$$w_1 = \{a_1, a_2, a_3, .., a_m\}$$

$$w_2 = \{b_1, b_2, b_3, .., b_m\}$$

WG2Vec then constructs a new sentence embedding by appending the two word embeddings, as opposed to the Paragraph2Vec algorithm which concatenates or averages them (see Section 2). More specifically, the sentence embedding for this wordgraph np_1 is now of length 2m:

$$np_1 = \{a_1, a_2, a_3, ...a_m, b_1, b_2, b_3, .., b_m\}$$

This is an example of a wordgraph consisting of two words. Similarly for a wordgraph with a single word a vector is constructed with a vector of length m. Based on expert knowledge, the following wordgraphs were used:

1. specific: each noun with an adjective
2. activity: main verb with object (noun)
3. leading: object of the main verb
4. noun: every noun in the sentence
5. verb: every verb in the sentence

Both the *specific* and the *activity* consist of two words, the others are one word per wordgraph. In the domain of skills matching, a recruiter will only add adjectives if they add to the noun (according to the domain expert, adjectives are rarely added to spice up the text). For instance, the skill "Python programming" differs greatly from "neuro-linguistic programming". The activity wordgraph is based on the assumption that the essence of a skill is performing an action, a verb, on or with an object, the noun. Different domains might require different wordgraphs however, this does not change the algorithm.

3.3 Defining Sentence Similarity using WordGraph2Vec

The previous subsection illustrates how texts are transformed into a numeric vector by WG2Vec. The varying lengths of the vectors require a different approach when comparing them to each other (to determine semantic similarity). Here we describe a similarity function that can deal with the proposed (variable length) wordgraphs. One such similarity function for WG2Vec is denoted in Algorithm 1. This algorithm takes two (simplified) sentences as input that will be transformed into wordgraphs, as described in Section 3.1.

The similarity is then based on exact (textual) matches of parts, with the so-called *overlap score* (see lines 6 and 7), as well as semantic similarity on different parts with the *difference score* (see lines 8 to 17).

Different elements are vectorized as described in 3.2. Cosine similarity gives the similarity (see line 15) for wordgraphs of the same type (and therefore of the same length). Note that, we provided a kernel K per wordgraph, which allows us to weigh the different words. For example, in the activity (i.e., a related verb and noun pair) the kernel may specify that the noun part is more relevant by multiplying the corresponding numbers in the vector with a specified value. A higher value would make that part more relevant when comparing. Per wordgraph, the addition of overlap- and difference score, weighted by the so-called Vector Kernel N, leads to the final score. To be more precise, the wordgraph kernel weighs the words in the wordgraph, whereas the vector kernel weighs the relative importance of vectors. Once all (weighted) overlap- and difference scores are processed, the final score gives the similarity between the two sentences. Higher numbers represent better matches.

4 Experimental Study

In this section we will elaborate on our learning task, i.e., finding matching skills between vacancy texts and a taxonomy. We will then provide more details on the data and algorithms used in the experiments. Next, we will describe the metrics used for benchmarking the algorithm. We conclude with the experimental results.

Algorithm 1 Similarity scoring algorithm used by WordGraph2Vec.

Require: Word2vec $w2v$, Wordgraph Types W, Wordgraph Kernel K, Vector Kernel N

function WG2V_SIMILARITY(Sentence S_1, Sentence S_2)
  $final\_score \leftarrow 0$
  For each word_graph_type $w \in W$ **do**
    $A \leftarrow S_1[w]$
    $B \leftarrow S_2[w]$
    $overlap\_score \leftarrow |A \bigcap B|$
    $asymmetric\_A \leftarrow (A - B)$
    $asymmetric\_B \leftarrow (B - A)$
    $difference\_score \leftarrow 0$
    For each word_graph $a \in asymmetric\_A$ **do**
      $score \leftarrow 0$
      For each word_graph $b \in asymmetric\_B$ **do**
        $\overrightarrow{V_a} \leftarrow w2v(a)$
        $\overrightarrow{V_b} \leftarrow w2v(b)$
        $score \leftarrow \max(score, cosine\_similarity(\overrightarrow{V_a}, \overrightarrow{V_b}))$
      end for
      $difference\_score \leftarrow difference\_score + score \cdot N_w$
    end for
    $final\_score \leftarrow final\_score + (overlap\_score + difference\_score) \cdot K_w$
  end for
  return $final\_score$
end function

4.1 Data

In the experimental study, we use two distinct sources of data. The first source is the taxonomy. The taxonomy is a handcrafted hierarchy maintained by a group of experts. It comprises different types of relevant concepts and abstraction levels. For the current research, only the lowest level text of a skill is used. Meaning the input to the algorithm from the taxonomy is a simple table with a skill identification and a text belonging to the skill. Some skills have the same texts, towards the algorithm this is simplified by only considering the unique skills resulting in 4676 skill descriptions. This is the set of data towards the algorithm needs to match.

The other source of data is the set of vacancies. This source differs greatly from the taxonomy. Firstly, the skills are less formal in their wording. Instead of describing the essence of the activity, it is often made more popular for instance "Programmeren in Python" (Coding in Python) would read "Je spreekt vloeiend Python" (You speak Python fluently)[3]. Secondly, the vacancies do not have a fixed predefined format for skills. This means they need to be extracted from the vacancy text before matching. This extraction is done by using a set of regular expressions. Resulting in fragments that should contain the part of the sentence with the skill description. Due to the nature of the extraction, it is possible that a sentence lacks important parts (e.g. the main verb of a sentence) and could be

[3] Literal translation.

unusable to the matching algorithm. Similar to the taxonomy items vacancies can describe a skill in an identical way to another vacancy. To only match a skill once, only the distinct texts are used resulting in a total of 84460 vacancy skills.

To check for the correctness of the algorithms a set of annotations is made by experts. These experts (the same as the ones maintaining the taxonomy) matched extracted skill texts to the taxonomy. They could mark a text as describing multiple skills (or unmatchable), not having a skill, or fitting to one taxonomy item. This resulted in a total of 1928 annotations of which a total of 626 were matched on taxonomy items. It is important to note that the users do not rank or quantify the match in any form (similar to the research done by Radlinski and Craswell [9]). Next to the limiting factors of the annotations, the number of unique classifications brings the total down to 612. This uniqueness is required as a skill could be presented multiple times to a different expert and could be matched to a different taxonomy item (with identical text).

4.2 Hyperparameters

The previously mentioned annotation count is but a fraction of the number of examples used to train BERT models[4, 10]. As such, training complex models on this data is currently not feasible. Due to the limited data source, in this study, both the initial setup (Algorithm 1) as well as the best matching algorithm are to be measured.

The WG2Vec algorithm has two main hyperparameters: Wordgraph kernel and Vector Kernel. These hyperparameters influence the weight of each wordgraph type and underlying vectors. For the wordgraphs an initial estimate was made, this can be seen in Table 1. This initial estimate involved running the algorithm against a sample of the data and adjusting the values accordingly. Later a random search for the best configuration of the hyperparameters was used [1].

4.3 Evaluation Metrics

To evaluate all algorithms equally a coarse metric is used to measure the quality. These metrics are based on the case of users wanting to minimize the amount of unmatchable and unusable skills and maximise the number of top K hits (where due to the user interface, the value of K is set to 5). Given the use-case there are multiple quality criteria:

1. Percentage of matchable skills
2. Precision@N[12]

Given that many algorithms have a fixed dictionary [6, 8, 10], it is important to measure how well the algorithms can fit the given words.

The first quality (Q_1) criterion is measured by counting the number of skills which can be used for the matching process divided by the total number of available skills. The value for Q_1 is computed by the formula depicted in equation 1.

algorithm nr.	1	74
Wordgraph Kernel		
leading	1.0	5.0
activity	14.0	9.0
specific	19.0	1.0
noun	5.0	0.0
verb	2.0	8.0
Vector Kernel		
leading	[1]	[1]
activity	[15,19]	[5,6]
specific	[16,14]	[7,9]
noun	[1]	[1]
verb	[1]	[1]

Table 1: Kernel values for the initial algorithm (1) and the best performing algorithm (74). The vector kernel is shown different from what is suggested in the algorithmic outline 1. As wordgraph vectors comprise one or multiple embeddings, each embedding is multiplied with its own weight. So with a wordgraph consisting of two embeddings each part is multiplied with its own scalar. This matrix shows these scalar values.

In this formula, the $||S||$ is the number of skills to be matched, with S being the set of skills and s as a specific skill. The function $V(s)$ is a check to see whether the given skill s can be used in matching.

$$Q_1 = \frac{\sum_s^S V(s)}{||S||} \tag{1}$$

This metric intends to get a view of how broad the number of acceptable skills is. If the percentage is high, this indicates the method can use more skills than if the percentage were lower.

The second criterion, Precision@N, is measured by looking into the fraction of the usable results. A result is usable if the match given by the user is present in the top k hits from the algorithm. As such the formula is as follows:

$$\frac{\sum_s^S ||\exists_p \{p \subset Top(P(s), k) | p = u\}||}{\sum_s^S V(s)} \tag{2}$$

Where P is the algorithms prediction function, the $Top(list, x)$ only takes the best k results from the $list$, and u is the annotation from the user. The function $V()$ is the same as in the first quality measurement. The k value for the research is set at 5 due to the user interface shown for evaluation to users. This metric intends to get a feeling for how good the predictions fit the user's expectations.

4.4 Experimental Results and Discussion

Using the previously defined criteria each algorithm is tested on the same data. The results of the tests can be seen in Table 2. The results were obtained by predicting the correct match.

Algorithm	Response ratio	Precision@5
WG2Vec [74]	$0,804\%$	$7,07\%$
WG2Vec [1]	$0,804\%$	$6,92\%$
USE	100%	$0,178\%$
sBert	100%	$3,08 * 10^{-2}\%$
Paragraph2Vec	100%	$1,18 * 10^{-3}\%$

Table 2: Results sorted by Precision@5 column.

The results show that the WG2Vec algorithm trades response for precision. The algorithm cannot find wordgraphs in all sentences and is further limited by the Word2Vec dictionary. Oftentimes these sentences are bad extractions done by the regular expressions (they lack the main verb). This results in a smaller set for which the algorithm can find matches compared to the other techniques. When it comes to the Precision@N of the algorithm the WG2Vec outperforms the others. Here it can be observed that the algorithm if it can find matches, has a better understanding of what is important to the user.

By looking into specific responses, the differences between the methods become clearer (shown in Table 3). This table shows that the responses given by Paragraph2vec appear to be almost random (not a single word meaning appears to fit). sBert performs better, some results do make sense but many appear almost as random as Paragraph2vec. This might be due to missing words in sBert's vocabulary combined with its attention being focused on the wrong parts of the sentence. USE performs better than the aforementioned methods. The top hit of USE is rarely random, albeit often not the correct item. The main issue with USE is in identifying the essence of the skill. This issue is shown in the first example, the sentence describes transporting liquid, and USE finds a match with transportation across water. The link might seem relevant, but the implications are different (one requires a drivers license, the other a boat license).

Vacancy tekst	Je vervoert immers vloeibare lading
WG2Vec [74]	Zware ladingen verhuizen
USE	Zware vrachtstukken over de binnenwateren vervoeren
sBert	Gevaarlijke stoffen vervoeren
Paragraph2Vec	Adviseren over planologie en volkshuisvesting
Vacancy tekst	Je toetst aanvragen in het kader van bouw wet- en regelgeving
WG2Vec [74]	Klanten juridisch advies geven
USE	Toegang tot archieven regelen rekening houdend met wet- en regelgeving
sBert	In een team werken
Paragraph2vec	Producten na bereiding verpakken
Vacancy tekst	Je komt het afval ophalen
WG2Vec [74]	Gevaarlijk afval sorteren en afvoeren
USE	Afval vernietigen
sBert	Toezichthouden op veiligheid en kwaliteit
Paragraph2Vec	Omlijstingen maken

Table 3: Best skill matches performed by the different algorithms.

The examples shown in Table 3 also show the complexity of the task. Finding the right match between skills in the different sources also requires knowledge of the implications from the text. If enough data would be present on this mapping, a machine learning algorithm could be used to build an embedding of these implications.

5 Conclusions and Future Work

In this paper, we proposed a novel algorithm called WG2Vec. The goal of the algorithm is to improve matching skills to an existing taxonomy. The algorithm dissects a sentence to a set of wordgraphs using dependency tree parsing and part-of-speech tagging. For each wordgraph, it creates vectors using Word2Vec. Using a comparison between the wordgraphs from a skill and an item in the taxonomy a match is made.

The metrics do not show a clear winner for every scenario. Although WG2Vec has a higher number of valid matches, the number of sentences usable for the method is relatively low. More research is needed to improve it to acceptable values. The large difference in the Precision@5 metric does show that the state-of-the-art can be improved upon by specifically engineered methods. For the current scenario of finding the highest number of valid hits, WG2Vec does a better job than the competition.

For future iterations, the underlying word embedding model could be substituted by the USE algorithm. Given that it is the best alternative method it could improve the Precision@5 score of the algorithm.

Instead of the more dedicated feature engineering, a BERT model could be trained to the data. By letting the BERT model try and predict the taxonomy item it could learn a more relevant representation. It is important to note that this would require more data than is currently available.

Another significant improvement would come from altering the initial step of sentence selection. By giving more complete sentences to the algorithm it (or the alternatives) might get a better Precision@5 score.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. **13**(2) (2012)
2. Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., Specia, L.: SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). pp. 1–14. Association for Computational Linguistics, Vancouver, Canada (August 2017). https://doi.org/10.18653/v1/S17-2001
3. Cer, D., Yang, Y., Kong, S.y., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.H., Strope, B., Kurzweil, R.: Universal Sentence Encoder (April 2018), http://arxiv.org/abs/1803.11175

4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (May 2019), http://arxiv.org/abs/1810.04805
5. Le, Q.V., Mikolov, T.: Distributed Representations of Sentences and Documents (May 2014)
6. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space (September 2013), http://arxiv.org/abs/1301.3781
7. Montani, I., Honnibal, M., Honnibal, M., Van Landeghem, S., Boyd, A., Peters, H., McCann, P.O., Geovedi, J., O'Regan, J., Samsonov, M., Altinok, D., Orosz, G., De Kok, D., Kristiansen, S.L., Miranda, L., Bot, E., Roman, Baumgartner, P., Fiedler, L., Hudson, R., Kannan, M., Edward, Howard, G., Phatthiyaphaibun, W., Tamura, Y., Bozek, S., Murat, Daniels, R., Flusskind: Explosion/spaCy: V3.4.1: Fix compatibility with CuPy v9.x (July 2022). https://doi.org/10.5281/ZENODO.1212303, https://zenodo.org/record/1212303
8. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. pp. 1532–1543 (2014)
9. Radlinski, F., Craswell, N.: Comparing the sensitivity of information retrieval metrics. In: Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '10. p. 667. ACM Press, Geneva, Switzerland (2010). https://doi.org/10.1145/1835449.1835560
10. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (August 2019), http://arxiv.org/abs/1908.10084
11. Statistics Netherlands: Vacatures. https://www.cbs.nl/en-gb/visualisations/labour-market-dashboard/vacatures?sc$_{l}ang = nl - nlsc_{i}temid = 39df8938 - f4b9 - 493b - 8d13 - 309a51e6f4f2(July$ 2022)
12. Yamashita, M., Shen, J.T., Ekhtiari, H., Tran, T., Lee, D.: JAMES: Job Title Mapping with Multi-Aspect Embeddings and Reasoning (Feb 2022)