Defining Multi-Word Expressions in Context: An Implementation Perspective

Tarek Mehrez

March 22, 2015

Contents

1 Introduction **3**

2 Theoretical & Linguistic Background **3**

3 Method **4**

3.1 Definitions & Initial Tools 4

3.2 Corpus 5

3.3 Implementation 5

3.3.1 MWE Annotation 5

3.3.2 Compile IOB File 5

3.3.3 Reformatting text 6

3.3.4 Constructing phrases 6

3.3.5 Measuring Distance 6

3.3.6 The Bigger Picture 7

3.4 Measuring Contexts 7

3.4.1 Baseline Calculation 8

3.4.2 Vector Extraction 9

3.5 Directory Structure 9

4 Results & Discussion **9**

5 Conclusion **11**

1 Introduction

In this work we investigate the definition of MWEs. We tried to introduce a scientifically-proven method to support our hypothesis in how can we define a MWE. Our hypothesis is concerned with contexts in which words may appear. Specifically, the contexts in which a MWE its constituents may appear. By definition those contexts should be different for the MWE to have its special meaning, that made it necessary to use in the first place. Linguistically, definitions may differ and even contradict when it comes to such a topic. Since the very beginning people had different definitions as we will see in later sections. Therefore, we decided to focus on a certain subclass of MWEs as shown later. To that end, we implemented a tool that measures contexts and their differences when it comes to a MWE and its constituents. This work is counted as a pre-requisite to complete the course "Detecting and classifying multi-word expressions", and not as a novel endeavour or a scientific suggestion to prove. But that doesn't deny the fact that further discussions may emerge from the results we introduce in this paper, hoping that we can contribute to a more concrete definition of MWEs.

Section 2 contains a brief linguistic and theoretical background on the general definitions of MWEs, but an extensive linguistic description of this work is shown in its twin document [5]. The details of the implementation are explained in section 3. Consequently, the results and conclusion are discussed in sections 4 and 5 respectively.

2 Theoretical & Linguistic Background

This work targets the definition of a multi-word expression (MWE). According to standard definitions, a MWE had different ways to be defined in terms of form an meaning. But sticking to the literal meaning, it would obviously mean an expression with multiple words or lexemes. What do these words represent or have in common is something that is not agreed on upon different researchers. According to [1] a MWE could be easily identified as an expression whose words are separated by white spaces. But they added a counter argument that disproves the whitespace approach. Other languages such as German has concatenated compound nouns that could be still defined as MWEs such as *Kontaktlinse* "contact lens".

A definition which is somehow straight forward is provided by [8], they argued that a MWE are a group of lexical items that consists of multiple constituents that represent some idiomaticity. We believe this is where different arguments may appear. Idiomaticity could be defined lexically, syntactically, semantically, pragmatically or statistically. This makes things a bit vague, as a concrete definition is still missing.

MWEs such as "living room" refer to a certain entity, and meet the first requirement by having multiple constituents. But whether the literal meaning is different or not, this is where the argument may emerge.

Different classes of MWE should be taken into consideration as well. For example named entities are a class of tokens that could intersect with the set of MWEs. By intuition not all named entities are MWEs, and vice versa. The named entity "San Fransisco", or "The United Nations" are definitely MWEs, but "England" is obviously not.

Although some references may consider "England" as a MWE [9], in this work this is simply ignored, as it contradicts the aforementioned definition.

One of the directions that have been taken is the difference in contexts. For example, idiomatic MWEs such as "cross the line" have some interesting characteristics that should be taken into consideration [8].Again, it

has been argued that for a group of tokens to form a MWE, they have to be idiomatic. Given that in the mentioned example the literal meaning is not the intended meaning behind this expression, it could appear in different context than the actual literal meaning [2].

This means that "cross the line" in a debating context, will be different than "cross the line" in a sports context. Same for the words "cross", "the" and "line" appearing in different contexts separately.

Other classes of MWEs such as compound nouns might also be interesting. The MWE "world cup qualifying matches" has a lot to say when it comes to contexts compared separately and as a whole, as we did previously. This MWE could refer to different contexts such as sports or politics, that's when we consider different combinations of tokens forming the whole MWE of course.

This of course doesn't apply to all types of MWEs. For example particle verbs won't be really interesting in that case [4]. The verb "find out" will not necessarily have different contexts when we break it down. Which brings us to the question whether we can consider particle verbs as MWEs or we can basically ignore context as a main factor of defining MWEs when it comes to some subclasses.

Nevertheless, the aforementioned argument can still prove that one of the main necessities to define MWEs, or at least a certain subclass, would be defining the contexts in which they appear, separately and as a whole.

And for that reason the main goal behind this work is to investigate the difference in contexts between the MWE as a whole, for example "world cup", and the contexts of its constituents "world" and "cup".

3 Method

This section explains the technical details of the method used in order to extract MWEs, identify their contexts and measure discrepancies. To that end a Java tool was implemented. The tool handled pre-processing alongside the contexts measurements. Further details are explained later in this section. As mentioned before, the twin document [5] of this work contains a more linguistic perspective on what has been done.

3.1 Definitions & Initial Tools

The technical implementation that can prove our hypothesis was somehow tricky. A concrete definition of a context was the hardest. Different works have tackled the idea of a context, and how can we divide a piece of text into different contexts or clusters according to word co-occurrences.

We start by mapping the linguistic hypothesis to the technical terms that we will use throughout this section. First of all we define the context to be the word clusters produced by k-means algorithm. Context windows of 5 words were used to calculate those clusters, given the number of clusters. In order to measure how far are these contexts, we used euclidean distance between word vectors. The vectors notion with MWes had been used before in other works such as [3].

Two external open-source tools were used to aid the vector calculation and clustering processes.We used the tools word2vec and glove to produce the vectors for the lexicon. Clusters were produced on top of the vectors using word2vec.

3.2 Corpus

To prove our hypothesis, we need a corpus that already provided manual annotations of MWEs to avoid the error propagated by automatically identifying them. Therefore, we decided to use the wiki50 ccorpus [9].

The corpus has 50 scientific articles as text files, and the respective MWE annotations as the MWE type, and the starting and ending character index in the text file. Also a compiled list of all MWEs in the corpus was available.

We combined all text files in one to apply the vector extraction and clustering. The training file has more than 95K tokens, with a distinct vocabulary size of about 15K tokens. A huge drawback was the lack of enough data as the number of tokens were not enough to train a large-scale model. But as we were also constrained by finding a well-annotated MWE corpus, we decided to stick to the wiki50 corpus

Limitaitons of the corpus and possible enhancements are discussed in section **??**.

3.3 Implementation

The implemented tool is a combination of several steps. With the help of the vector-calculation tools word2vec and glove, we were able to construct a pipeline where all intermediate steps could be treated as a black box. That was done on purpose so that any of the separate steps could be used externally.

The tool that handled the main traffic was implemented in Java and could be accessed via the following link `https://github.com/tarekmehrez/MWE-in-Context`. To run the tool, one can pass any of the possible parameters to that jar file `mwe.jar`. Possible parameters and their respective usage are described in detail in the upcoming subsections.

3.3.1 MWE Annotation

The wiki50 MWE annotations were marked with the character index within the text file. So we had to produce an actual list of MWEs for each file using both the text and the annotations. The first option `--annotate-corpus` takes the root directory to the text and annotations, to produce the actual MWE list for each text file. Luckily enough, the corpus had a compiled list of all MWEs in all files, so instead of the implemented option, we could use the compiled list. But this option is kept in case we needed separate annotations for each file.

3.3.2 Compile IOB File

The compiled list had all tokens in the text, including non-MWEs. So the implementation included an option that takes this file, to produce one last MWEs list alongside their respective types. The parameter `--compile-iob` could be used when running the jar file, with the path to the list provided by the corpus (the iob file). This parameter produces a list of all MWEs and their types separated by commas. The entries of this file were pre-processed by removing special characters and lowercasing the MWEs to match the text pre-processing described later. This list helped producing specific statistics about the results for each separate type. A snapshot of the compiled list is represented in the following figure.

```
military air traffic control,MWE_COMPOUND_NOUN
free falling seal team,MWE_COMPOUND_ADJ
flying aircraft,MWE_COMPOUND_NOUN
military aircraft,MWE_COMPOUND_NOUN
touch and go approach,MWE_COMPOUND_NOUN
combat gear,MWE_COMPOUND_NOUN
makes a quick decision,MWE_LVC
emergency cord,MWE_COMPOUND_NOUN
reserve chute,MWE_COMPOUND_NOUN
roman catholic,MWE_COMPOUND_ADJ
```

Figure 1: Snapshot of the compiled MWEs list

3.3.3 Reformatting text

As we treated the whole corpus as one big text file containing all articles, we had to pre-process this file to make sure it's suitable for further processing. The option `--reformat-text` was implemented to take the text file, lowercase it, remove special characters and write the results in a new file. That was necessary to make sure that no special characters can intrude the extraction of MWEs or matching them for further steps such as vector extraction.

3.3.4 Constructing phrases

In order to produce vectors for the whole corpus, we need to treat each MWE as one token or entry, so that we can have a special vector for this exact MWE. This vector is to be compared to its separate tokens later on. Therefore, we had to implement and option that takes both the text file, and the compiled MWE list. The parameter `--construct-phrases` produces the same exact text file entered, but this time with the MWEs marked by underscored between its tokens. So for example the term "world cup" appearing in the initial text file will appear as "world_cup" in the resulting file.

3.3.5 Measuring Distance

The main module in this tool is the last one. The parameter `--compute-distances` takes the file containing the vectors, the compiled MWE list and the clusters file, to produce the final results.

The results file contains all MWEs, their types, the euclidean distance between them and their constituents, and the baseline for each MWE. Finally, it contains the results for each MWE type including the number of correctly classified instances, or those who had different contexts than their constituents, and the final accuracy.

The next figure shows a snapshot of one of the results files.

Vector extraction methods and baseline calculations are described in the upcoming subsection.

```
mwe,type,distance,baseline
gained_international_attention,gained_international_attention,MWE_LVC,1.1784566162370547,4.351922229059476
military_academy,military_academy,MWE_COMPOUND_NOUN,1.772206392312278,4.8219982195036115
adult_education_classes,adult_education_classes,MWE_COMPOUND_NOUN,1.4376989571042174,5.21016089452562
social_outcast,social_outcast,MWE_COMPOUND_NOUN,0.8327182981696151,8.233731883128815
housewarming_party,housewarming_party,MWE_COMPOUND_NOUN,1.4106626763318542,5.1096660831737335
draws_his_main_support,draws_his_main_support,MWE_LVC,2.6497980233710856,5.109857980523138
.
.
.
################################
Number of MWEs: 2705
Accuracy for each MWE type:
MWE_COMPOUND_ADJ: 34 out of 76 with 44.74\% accuracy
MWE_IDIOM: 17 out of 18 with 94.44\% accuracy
MWE_OTHER: 13 out of 21 with 61.9\% accuracy
MWE_LVC: 224 out of 361 with 62.05\% accuracy
MWE_COMPOUND_NOUN: 1810 out of 2848 with 63.55\% accuracy
Total accuracy of difference in contexts: 79.19\%
```

Figure 2: Snapshot of the results file

3.3.6 The Bigger Picture

Now that we had all pieces, we can just formulate the whole procedure by defining one big pipeline of steps. Example commands on how to run each step are available in the running script `run.sh`.

1. All text files are compiled in one and processed as described above
2. All MWEs are gathered in one list with their types, after being processed
3. MWEs are matched in the text files, to produce vectors for MWEs as a whole
4. K-means implementation in word2vec is applied to cluster the vocabulary.
5. Vectors are extracted using word2vec and glove
6. Vectors, clusters & MWEs list are fed back to that tool to measure discrepancies and calculate results

Figure 3 describes this implemented pipeline.

3.4 Measuring Contexts

As discussed before, contexts were basically clusters calculated according to context-windows. And in order to measure differences between clusters, we had to calculate euclidean distances between vectors.

So for each MWE to pass the test, it must meet the following criteria:

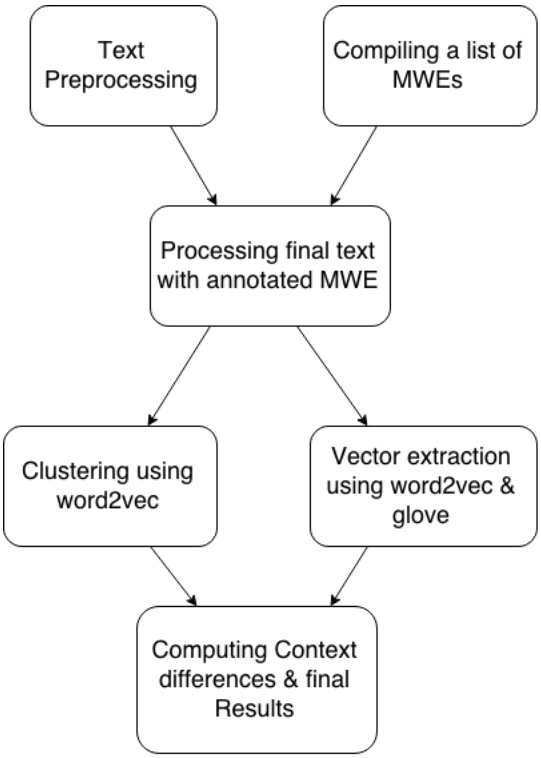

Figure 3: Snapshot of the compiled MWEs list

1. The distance between the MWE's vector and the average vector of its constituents must be greater than its baseline
2. The MWE's vector must be different from all vectors of its constituents
3. The MWE must lie within a different cluster than its constituents

Then and only then, a MWE will pass the criteria and is said to have different context than its constituents.

3.4.1 Baseline Calculation

Calculating the baseline was a bit tricky, since we needed an evaluation metric for our claim to make sense and prove our hypothesis. Considering that we have word vectors as our main method of calculating contexts, we calculated our baseline to be the maximum distance between the MWE and all other phrases occurring in the same cluster. That way we ensure that this distance represent the edge of the cluster, and that any greater distance should naturally occur out of the cluster, and therefore be in another context. To that end, we calculated for each MWE its own baseline to be compared with the average distance of its constituents as explained before.

3.4.2 Vector Extraction

To extract vectors we used the open-source tools word2vec and glove as mentioned before. The main difference is the implementation of both. Glove extracts the vectors by following a global training approach [7]. Basically a co-occurrence matrix is constructed following any global matrix factorization method (e.g Latent Semantic Analysis), then further training is done globally rather for a certain context. This global training considers the whole data set in each iteration rather than just a certain window.

On the other hand context window methods that are implemented in word2vec, focuses on a certain context instead of the whole data set. Examples of context window models that are implemented in word2vec are the continuous bag of words models (CBOW) and the skip-gram model, both are described in [6].

All methods were used to investigate different results in ourt task.

3.5 Directory Structure

The directory structure is divided as follows

- src: the java source code
- libs: external tools for the java application
- corpus: the wiki50 corpus
- doc: documentation
- tools: external tools (word2vec & glove)
- vectors: the produced vectors and classes/clusters
- results: results files

4 Results & Discussion

The results were pretty promising as they proved our hypothesis, at least in our controlled environment.

We tested the system with all types of vectors (cbow, skip-gram & glove) and different number of clusters (10,50,100,250,500).

The results are shown in table 4.

As the results look a bit optimistic, we may assume that a lot of factors may affect the reliability of this approach. First of all the corpus didn't have enough data as we mentioned before, also maybe the scientific domain is not the best one to test difference in contexts on. But again as we were constrained with an annotated corpus, wiki50 seemed like a good candidate.

Concerning the method, the mentioned criteria did a lot of assumptions on how vectors and clusters represent contexts, but that's not necessarily true as the relation between vectors and clusters wasn't really there.

Clusters	Accuracy
10	79.19%
50	95.19%
100	97.82%
250	98.19%
500	99.3%

Table 1: Results with different number of clusters

Maybe a good idea would have been linking clusters to vectors by having an actual hierarchy of clusters.

As for using different types of vector extraction tools, that turned out to be unnecessary as the results were surprisingly the same for cbow, skip-gram and glove given the differences between their implementations.

One major drawback was the huge loss in the number of MWEs from the original compiled list and the produced vectors. Initially the number of MWEs extracted from the corpus was 3324, but the vector-extraction tools produced vectors for just 2705. That loss in the number of MWEs could have made a difference in the results.

Also further enhancements could be further pre-processing, including stop word removal (except those who appear in MWEs) and some stemming to decrease number of vectors for similar words.

Concerning the implementation of the k-means algorithm, we did further investigation on how good did extract intuitive contexts. To that end some of the MWEs were selected to visualize their extracted contexts. Figures 4 and 5 show context differences between the MWEs "away_fans" and "taking_charge" and their constituents with number of total clusters equal to 500.

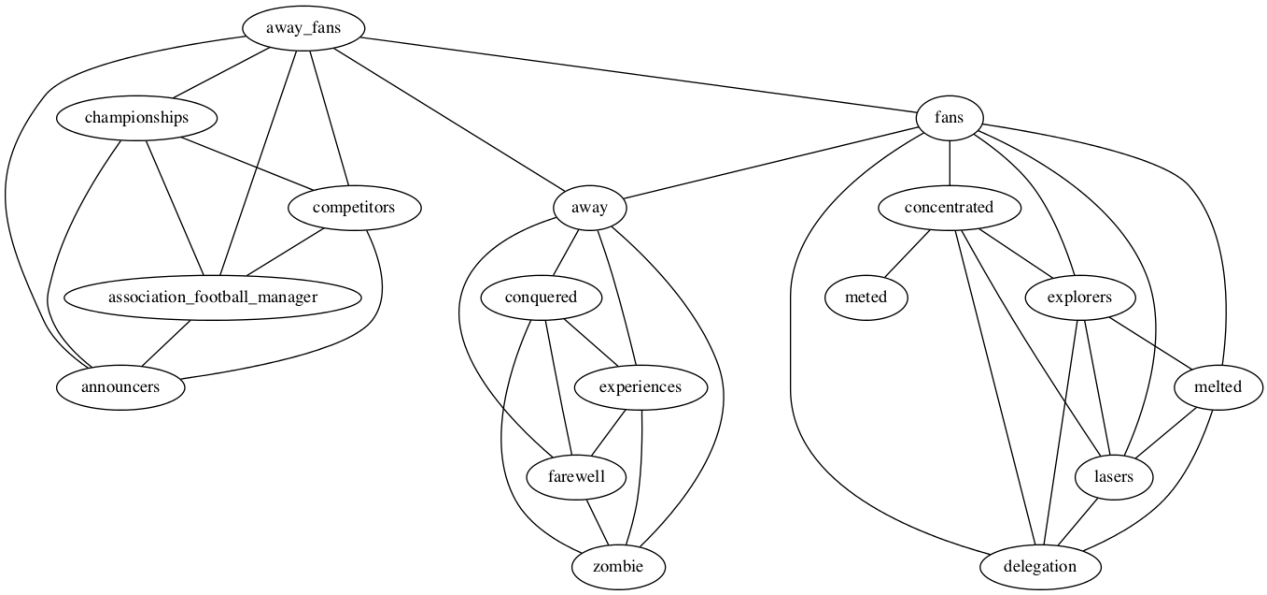

Figure 4: Distribution of words for the MWE away_fans

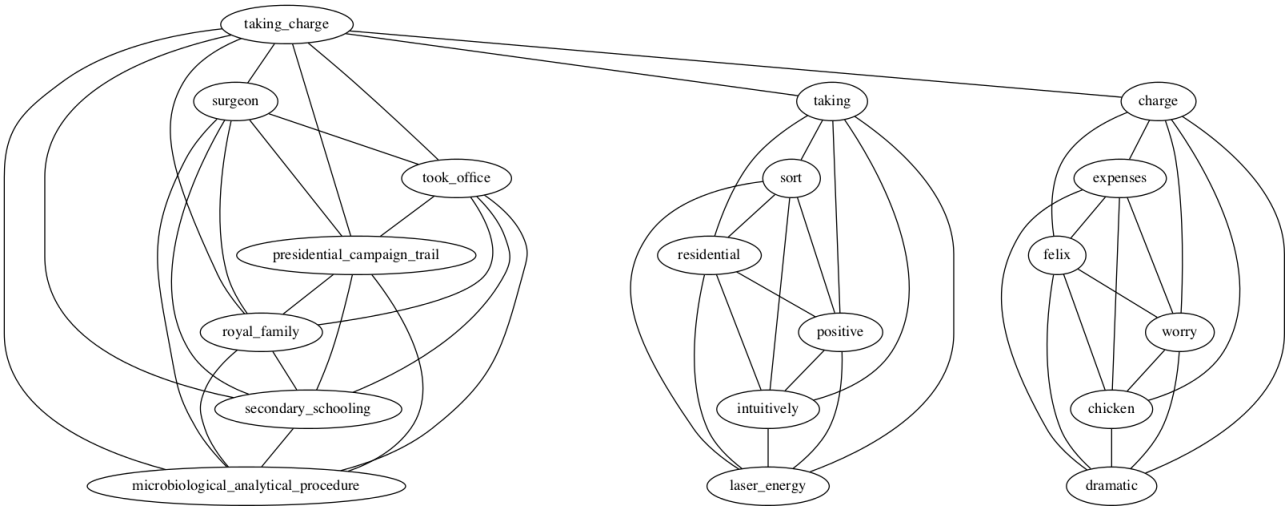

Figure 5: Distribution of words for the MWE taking_charge

Words in the previous figures are manually selected, to give a glance of the word distributions within the clusters. As shown, words within the same cluster don't have one concrete meaning or context, given that the number of clusters is 500. That's why some of the co-occurring words had slightly different relevance or no relevance at all.

Nevertheless, those figures still prove the hypothesis given that the contexts don't really overlap.

5 Conclusion

We presented in this paper a hypothesis that claims that a certain subclass of multi-word expressions appear in different contexts than their constituents. That was motivated by the different definitions of MWEs that didn't really agree on one clear definitions.

As a contribution towards a better definition, we implemented a tool that measures contexts of MWEs and context discrepancies.

Our results showed that there is a lot of potential behind this idea, given that the non-primitive criteria produced promising results.

In the future, incorporating better topic-extraction or clustering algorithms will produce better and more reliable results.

References

[1] Timothy Baldwin and Su Nam Kim. Multiword expressions. *Handbook of Natural Language Processing, second edition. Morgan and Claypool*, 2010.

[2] Nicoletta Calzolari, Charles J Fillmore, Ralph Grishman, Nancy Ide, Alessandro Lenci, Catherine MacLeod, and Antonio Zampolli. Towards best practice for multiword expressions in computational lexicons. In *LREC*, 2002.

[3] Graham Katz and Eugenie Giesbrecht. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 12–19. Association for Computational Linguistics, 2006.

[4] Francesca Masini. Multi-word expressions between syntax and the lexicon: the case of italian verb-particle constructions. *SKY Journal of Linguistics*, 18(2005):145–173, 2005.

[5] Tarek Mehrez and Anna Konobelkina. Defining multi-word expressions in context: A linguistic perspective. Technical report, Institute for Natural Language Processing, University of Stuttgart.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[7] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12, 2014.

[8] Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. Multiword expressions: A pain in the neck for nlp. In *Computational Linguistics and Intelligent Text Processing*, pages 1–15. Springer, 2002.

[9] Veronika Vincze, István Nagy, and Gábor Berend. Multiword expressions and named entities in the wiki50 corpus. In *RANLP*, pages 289–295, 2011.