Extracting Multi-Word Expressions Representing Technical Terms and Proper Nouns in Log Messages

Kilian Dangendorf, Sven-Ove Hänsel, Jannik Rosendahl,
Felix Heine, Carsten Kleiner, Christian Wartena
Institute for Applied Data Science Hannover (Data|H)
Hochschule Hannover
{kilian.dangendorf, sven-ove.haensel, jannik.rosendahl,
felix.heine, carsten.kleiner, christian.wartena}@hs-hannover.de

Abstract

IT-systems generate log messages containing important information about the system's health. To gather information about system entities, we extract multi-word expressions (MWEs) representing technical terms and proper nouns from a wide range of log messages from 16 different real systems. We apply Gries' information-theoretic approach which iteratively calculates the best MWE candidates using an eight-dimensional ranking method. These candidates are evaluated in an annotation study, achieving a precision of 66 %. This value is significantly higher than evaluations on general-purpose texts, demonstrating the higher occurrence of compound technical terms and proper nouns in log messages. The MWEs found can be used to reduce the number of nodes in a system behavior graph while increasing the information density of the nodes.

1 Introduction

Every computer system generates log messages. Analyzing and interpreting these messages allows conclusions to be drawn about the health of the system and possible cyberattacks. Log messages are intended to be read by developers who are familiar with the computer domain and therefore contain names of system parts, numbers, dates, memory addresses, etc. Logs are often short incomplete sentences (e.g. in most cases the subject is missing) with few or no verbs but many technical terms (see Listing 1 for examples). Thus, multi-word expressions (MWEs) make up a larger proportion than in general English texts.

This work forms the basis for our subsequent research, in which we construct a behavior graph from log messages. In the first modeling step, each token represents a node in the graph, resulting in large graphs. The more nodes we can eliminate from this point onwards while retaining all relevant information, the more efficiently the downstream graph neural network (GNN) can work, since there will be fewer nodes and higher information density.

Established methods for finding MWEs can help, as several tokens can be merged into one that is more descriptive. Technical terms and proper nouns in log messages are valuable because they describe or reference an entity in the system. This allows us to reduce the number of tokens as well as specify the systems entities more precisely.

The concepts of log parsing and current work in the applications of NLP on log messages as well as MWE extraction methods are described in Section 2. Section 3 shows how we process log messages and extract MWEs as technical terms and proper nouns. The results of our annotation study are presented in Section 4. Finally, we conclude our findings in Section 5.

2 Related Work

Listing 1: Example log lines from the Linux dataset containing month, date, time, log level, logging component and the log message (bold).

```
Oct 23 12:40:02 combo cups-lpd[22514]:
 ↪ Unable to get command line from client!
Oct 25 10:08:47 combo kernel:
 ↪ Inode-cache cache hash table entries: 16384 (order: 4,
 ↪ 65536 bytes)
Nov 22 14:31:58 combo kernel:
 ↪ httpd: page allocation failure. order:0, mode:0x1d2
Nov 22 14:32:24 combo kernel:
 ↪ sendmail: page allocation failure. order:0, mode:0x1d2
Dec  6 12:22:57 combo kernel:
 ↪ PID hash table entries: 512 (order: 9, 4096 bytes)
```

2.1 Log Parsing

System logs are notoriously unstructured and often include metadata such as timestamps, log levels, or logging components. Parsing these raw logs into their building blocks, metadata, and log templates by identifying static and variable parts is a common method. Zhu et al. (2019) provide an open-source toolkit to compare the accuracy of 13 different log parsers on a benchmark dataset, which was later published independently as *Loghub* (Zhu

Proceedings of the 22nd Workshop on Multiword Expressions (MWE 2026), pages 61–65

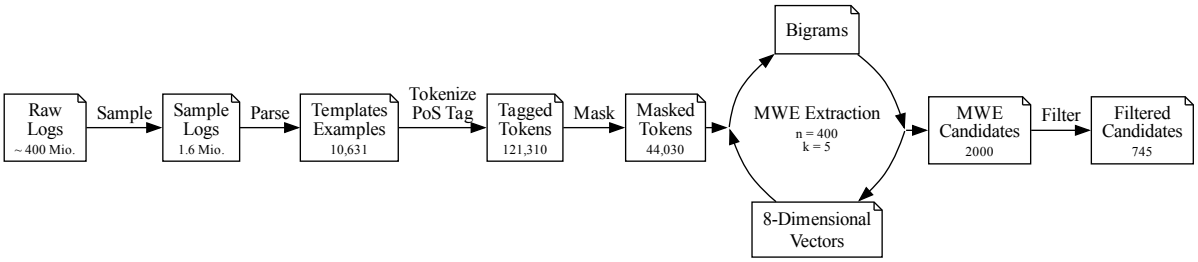

Figure 1: Our automated data processing pipeline applying Gries' MWE extraction method.

et al., 2023). The aim is to correctly assign each log message to the corresponding template.

Listing 1 shows some example lines of the Linux dataset with metadata, the message itself is highlighted. Splitting metadata and messages from Listing 1 and parsing them leads to the templates shown in Listing 2, where each variable part was replaced by <*>. Large amounts of log messages can be reduced to a small number of templates. E.g., the Linux dataset consists of 25,000 log messages, which result in 488 templates (Zhu et al., 2019).

Listing 2: Templates resulting from Listing 1. Underlined parts indicate the desired MWEs.

```
Unable to get command line from client!
<*> hash table entries: <*> (order <*>: <*> bytes)
<*>: page allocation failure. order:<*>, mode:<*>
```

2.2 NLP on Log Messages

NLP techniques are increasingly being used in log parsing. *PosParser* (Jiang et al., 2024) uses PoS tags to identify variable and static parts of log messages and thus infer the corresponding template. The sequence of all verbs and the first noun play a key role here. A similar approach is that of *NLog* (Pi et al., 2019) in which all nouns are selected as key candidates and filtered by frequency in the dataset. The tool *NERLogParser* (Studiawan et al., 2018) uses deep learning techniques to identify and extract named entities from log messages. Shikha and Timalsina (2022) use a similar deep learning based named entity recognition (NER) method to find templates in log messages.

Lv et al. (2021) use PoS tags to remove *unnecessary* words, and apply a multi-layer LSTM to the embeddings of the remaining words to detect anomalies. Li et al. (2022) use PoS tagging and NER to generate a template vector again using Word2vec that is used as input to a deep neural network to find anomalies.

2.3 MWE Extraction

EXTERLOG is a tool for terminology extraction from log files (Saneifar et al., 2009a,b). Co-occurrences are extracted using PoS tags and the resulting syntactic patterns. Pointwise mutual information (PMI) and the Dice coefficient are used as ranking metrics. Gries (2022) uses 8 dimensions, representing different, most information theoretic properties of MWEs, in an iterative apporach.

Ben Youssef (2024) evaluated resulting MWE candidates and reported a precision of 48% on English texts. Bagdasarov and Teich (2025) demonstrate that Gries' method can be transferred to German and report a precision of 61 %.

3 From Logs to MWE Candidates

We use the 16 datasets from Loghub. They represent a wide range of software systems, such as distributed systems, supercomputers, operating systems, mobile systems, server applications, and standalone software. In total, these datasets contain over 400 million log lines.

We sampled 100,000 random lines from each of the 16 datasets resulting in a total of 1.6 million log messages. To achieve a sample that is diverse and does not have many repetitions of almost identical log lines, we used the established log parser Drain (He et al., 2017) to extract log templates. For each unique template we select one random example log message. This prevents frequent templates from being given excessive importance. After sampling we have 10,631 log messages consisting of 121,310 tokens (see Figure 1).

To target technical terms, we pre-filter relevant word types. Saneifar et al. (2009b) achieved better results when they pre-filtered according to syntactic patterns of PoS. With POSLOG (Dangendorf et al., 2025), we have a tool that was developed specifically for tagging log messages[1]. POSLOG tags in Universal PoS-tags with 17 tags (Petrov et al., 2012; Nivre et al., 2016).

For technical terms and proper nouns, nouns, proper nouns, adjectives and adverbs are of special

[1] POSLOG also comes with a tokenizer that specializes in log messages, which we have also used here.

interest[2]. We mask tokens of other PoS so as not to destroy the sentence structure. When forming bigrams, masked tokens are skipped. For the MWE extraction, 44,030 tokens remain unchanged.

3.1 Identifying MWE Candidates

To extract the MWEs, we use the method of Gries (2022). This involves ranking all bigrams in several iterations, with the best candidates being merged into one token before the next iteration starts. This way, even long MWEs are found if they achieve a high ranking. For the ranking, an 8-dimensional vector is generated for each bigram ab:

Frequency (1) is the overall occurrence of the bigram, logarithmically dampened.

Dispersion (2) states how much the distribution of the bigram across the 16 datasets deviates from the distribution of the proportions of the respective datasets using Kullback-Leibler divergence (KLD).

Type Frequency (3 & 4) counts how many different types follow a and precede b, dampened logarithmically.

Entropy (5 & 6) calculates the normalized entropy after a and before b. Similar to dimensions 3 and 4, but here the frequency distribution of the following/preceding types is taken into account.

Association (7 & 8) gives the degree to which a attracts b and b attracts a (calculated by KLD).

If not already normalized, these dimension values are min/max normalized to the interval [0;1] for each iteration n. The dimensions are designed in such a way that good MWE candidates generate high values. In each iteration, the Euclidean distance to the origin of the hypercube is calculated. The k MWE candidates with the highest distance are selected as MWEs from the iteration and are merged. For our experiment, we choose n=400 iterations and k=5 and we get 2,000 MWE candidates.

Referring to the initial example in Listing 2, *hash table* was found in iteration 22 and *hash table entries* in iteration 47, *page allocation* in iteration 64 and *page allocation failure* in iteration 77, and *command line* in iteration 159.

3.2 Post-Filter

We dropped MWE candidates based on three rules: First, we filter out candidates whose tokens contain punctuation characters like URLs, paths and other tokens that are atypical for words (e.g., `ccfile::copyfile`, `krbtgt/#24#@#24#`). Second, we filter out words with more than 15 characters[3] (e.g., `ksserverupdaterequestdelegate`). Third, we exclude candidates whose last token does not have the PoS tag `NOUN` or `PROPN` (e.g., *too many*, *so far*). After filtering, we have 745 candidates.

[2]A technical term built from adjective and noun would be *public key* for example.

Table 1: Proportions of successful and unsuccessful MWE candidates with majority vote.

MWE Candidates	Count	Rel.
Unsuccessful	253	34 %
Successful	492	66 %
Total	745	100 %

4 Results

Three computer scientists annotated these 745 candidates. They annotated the extracted candidates as being perceived as a technical term in the domain of the dataset or not being a typical term in that domain. The annotators agreed on 486 candidates: 346 accepted and 140 rejected. This results in a Fleiss' kappa for all annotators of 0.49, which is in the middle range of moderate agreement. The deviation in the annotation can be explained by the fact that log messages are application-specific, and each annotator has a different background of experience. Additionally some phrases can be interpreted in multiple ways. The most common reason for agreement deviation were noun phrases consisting of adjective and noun. For example, *public key*, *floating point*, and *local host* are established terms in computer science and definitely belong together. Uncertain candidates include, for example *real time*, *excessive wakeups*, *read-only filesystem*, and *remote host*, where the annotators disagreed.

In the following we use the majority vote as gold standard. For 66 % of the extracted MWEs (see Table 1) the majority perceived the candidate as a real technical term or proper noun, which proves the effectiveness of the extraction method.

4.1 Classification

The annotators' second task was to classify the successful candidates, distinguishing between technical terms and proper nouns. The results can be found in Table 2. A total of 424 technical terms and

[3]An analysis of over one million English words found 136 words longer or equal to 15 characters. We therefore choose a limit of 15 characters to exclude long tokens that are unlikely to be words. Archived link: https://web.archive.org/web/20090427054251/http://www.maltron.com/words/words-longest-modern.html

30 proper nouns were identified. The proportion of proper nouns may seem small at first glance. However, we are dealing with MWEs, so single-token proper nouns such as *Linux* or *Google* are not included. The extracted proper nouns can be divided into three subclasses: We found *red hat linux* in the Thunderbird dataset as an example for a *company*, *dave jones* in the Linux dataset as an example for a *person*, and *internet systems consortium* in the Thunderbird dataset as an example for a *group*.

Table 2: Classification results on successful MWE candidates with majority vote. Parity arises when two annotators voted for successful, but different classes.

Class	Count	Rel.
Technical Terms	424	86 %
Proper Nouns	30	6 %
Majority Vote	454	93 %
Parity	38	8 %
Total	492	100 %

4.2 Most Widely Spread Examples

The term *file descriptor* appears scattered across most datasets: five times in four datasets in total. Spread across three datasets each there are the following exemplary terms: *configuration file* appears seven times, *network connection* six times, *exit status* also six times, and *lock file* three times.

The low number of the same technical terms across multiple datasets indicates that the language used in the respective systems is largely independent domain-specific terminology.

4.3 Occurrence in the Datasets

We found at least one MWE in every dataset sample. In total, we found 454 technical terms and proper nouns in 1,513 of the 10,631 (14.2 %) messages.

Combining these MWE tokens into one token saves 1,859 of the 121,310 total tokens. This corresponds to a savings rate of 1.5 %. The distribution across the datasets is shown in Table 3. While the Thunderbird dataset provides about a quarter of the MWEs in this work, only one MWE was found in the Apache dataset.

4.4 Longest Terms

The distribution of the successful MWEs candidates by their count of tokens can be found in Table 4. Almost 80 % of MWEs consist of two tokens, 15.6 % of three tokens, and 2.9 % of four. MWEs consisting of five or more tokens make up the remaining 1.3 %. Examples are *pci hot plug pci core* from Linux and *google software update installer* from Mac. The longest technical terms we found consist of six tokens coming from Thunderbird: *usb universal host controller interface driver*.

Table 3: Token replacement counts per dataset. For example, with the help of the successful MWE candidates, 490 tokens could be reduced in the Thunderbird dataset.

Dataset	Count	Rel.	Dataset	Count	Rel.
Thunderbird	490	26.36 %	OpenSSH	16	0.86 %
Android	361	19.42 %	HPC	13	0.70 %
BGL	342	18.40 %	Zookeeper	10	0.54 %
Mac	305	16.41 %	Proxifier	5	0.27 %
Linux	168	9.04 %	OpenStack	4	0.22 %
Hadoop	70	3.77 %	HDFS	3	0.16 %
Windows	52	2.80 %	HealthApp	2	0.11 %
Spark	17	0.91 %	Apache	1	0.05 %
		Total 1,859	100 %		

Table 4: Distribution of successful MWE candidates by their length in tokens.

Tokens	Count	Rel.
2	362	79.7 %
3	71	15.6 %
4	13	2.9 %
5	5	1.1 %
6	1	0.2 %
Total	454	100.0 %

5 Conclusion

Applying Gries' method on log messages results in a precision of 66 % for automatically finding technical terms and proper nouns, exceeding related work and indicating more MWEs in log messages.

Limiting the results to these two classes reduces the overall quality of finding MWEs in this evaluation. Without pre- and post-filtering this method also finds useful MWEs such as *no such file or directory*, *too many open files*, or *file not found*.

As only very few MWEs appear in more than three datasets, we conclude that logs use highly domain-specific terminology. In other words, it is to be expected that only a few of the MWEs extracted here will appear in a new dataset.

Through multiple iterations, MWEs consisting of up to six tokens were found. By combining all successful MWEs candidates in the entire data selection, we can reduce the number of tokens by 1.5 %. It should be noted that we have sampled the data by templates, so this value may vary when counted across all messages. However, this takes us a step closer to our goal of reducing the number of tokens and specifying the system entities more precisely.

References

Sergei Bagdasarov and Elke Teich. 2025. Applying an information-theoretic approach for automatic identification of German multi-word expressions. In *Proceedings of the 21st Conference on Natural Language Processing (KONVENS 2025): Long and Short Papers*, pages 295–305, Hannover, Germany. HsH Applied Academics.

Chadi Ben Youssef. 2024. *mMERGE: a corpus driven Multiword Expressions discovery algorithm*. Ph.D. thesis, UC Santa Barbara. ProQuest ID: BenYoussef_ucsb_0035D_16720. Merritt ID: ark:/13030/m5457324. Retrieved from https://escholarship.org/uc/item/4ph4517b.

Kilian Dangendorf, Sven-Ove Hänsel, Jannik Rosendahl, Felix Heine, Carsten Kleiner, and Christian Wartena. 2025. PosLog: Creating a Part of Speech Tagger for Log Messages. In *2025 IEEE 13th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, pages 1444–1449.

Stefan Th Gries. 2022. Multi-word units (and tokenization more generally): a multi-dimensional and largely information-theoretic approach. *Lexis*, (19).

Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. 2017. Drain: An Online Log Parsing Approach with Fixed Depth Tree. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 33–40.

Jinzhao Jiang, Yuanyuan Fu, and Jian Xu. 2024. Posparser: A heuristic online log parsing method based on part-of-speech tagging. *IEEE Transactions on Big Data*, pages 1–12.

Zezhou Li, Jing Zhang, Xianbo Zhang, Feng Lin, Chao Wang, and Xingye Cai. 2022. Natural language processing-based model for log anomaly detection. In *2022 IEEE 2nd International Conference on Software Engineering and Artificial Intelligence (SEAI)*, pages 129–134.

Dan Lv, Nurbol Luktarhan, and Yiyong Chen. 2021. Conanomaly: Content-based anomaly detection for system logs. *Sensors*, 21(18).

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).

Aidi Pi, Wei Chen, Will Zeller, and Xiaobo Zhou. 2019. It can understand the logs, literally. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 446–451.

Hassan Saneifar, Stéphane Bonniol, Anne Laurent, Pascal Poncelet, and Mathieu Roche. 2009a. Mining for relevant terms from log files. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, pages 77–84. SciTePress - Science and and Technology Publications.

Hassan Saneifar, Stéphane Bonniol, Anne Laurent, Pascal Poncelet, and Mathieu Roche. 2009b. Terminology extraction from log files. In *Database and Expert Systems Applications*, pages 769–776, Berlin, Heidelberg. Springer Berlin Heidelberg.

Saloni Shikha and Arun Kumar Timalsina. 2022. Automated log parsing through named entity recognition. In *Proc. of the 12th IOE Graduate Conference*, volume 12, pages 1747–1753.

Hudan Studiawan, Ferdous Sohel, and Christian Payne. 2018. Automatic log parser to support forensic analysis.

Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu, and Michael R. Lyu. 2023. Loghub: A large collection of system log datasets for ai-driven log analytics. In *IEEE International Symposium on Software Reliability Engineering (ISSRE)*.

Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R. Lyu. 2019. Tools and benchmarks for automated log parsing. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '19, page 121–130. IEEE Press.