Abbreviation Expander - A web-based system for easy reading of technical documents

Manuel R. Ciosici
UNSILO A/S and
Aarhus University
Aarhus, Denmark
manuel@cs.au.dk

Ira Assent
Department of Computer Science
Aarhus University
Aarhus, Denmark
ira@cs.au.dk

Abstract

Abbreviations and acronyms are a part of textual communication in most domains. However, abbreviations are not necessarily defined in documents that employ them. Understanding all abbreviations used in a given document often requires extensive knowledge of the target domain and the ability to disambiguate based on context. This creates considerable entry barriers to newcomers and difficulties in automated document processing. Existing abbreviation expansion systems or tools require substantial technical knowledge for set up or make strong assumptions which limit their use in practice. Here, we present Abbreviation Expander, a system that builds on state of the art methods for identification of abbreviations, acronyms and their definitions and a novel disambiguator for abbreviation expansion in an easily accessible web-based solution.

1 Introduction

Abbreviations and acronyms are often used in text documents and denote typically long, often domain-specific, concepts that authors need to refer to multiple times. However, the use of abbreviations and acronyms can make reading and understanding difficult for people new to a specific field, can lead to confusion, and make automated text processing challenging, for example, in indexing text documents. Unfortunately, expanding abbreviations is a complex task. The meaning of some abbreviations and acronyms (e.g. DNA meaning *deoxyribonucleic acid* in biology-related domains) is often considered well-known, and is rarely defined in documents using them. Other abbreviations and acronyms can denote multiple concepts, depending on their context (e.g. PCB can refer to a number of distinct concepts[1]).

Available abbreviation expansion systems are limited in their usefulness either due to requiring technical knowledge on the user side or by relying on simple, dictionary-based methods which cannot be applied to ambiguous abbreviations that have more than one meaning. We present a system that automatically expands abbreviations and acronyms in a user provided document. Our system is a web-based application that does not require that users have experience setting up pipelines for Natural Language Processing. We build on state-of-the-art Natural Language Processing techniques and a novel disambiguation method based on unsupervised learning.

2 System Architecture

By building a web application, we aim to make users oblivious to the technical complexities of processing natural language. From a user's point of view, they upload a text file to the system and immediately see the file's content with all abbreviations and acronyms expanded.

Figure 1 shows the architecture of Abbreviation Expander's back-end. Text is first tokenized and split into sentences, after which a number of abbreviation expanders are used. Finally, their results are combined. The biggest part of our system is composed of the processing pipeline. We use the *UIMA*[2] framework as the basis for our system because it provides mature support for construction of processing

[1] https://en.wikipedia.org/wiki/PCB

[2] https://uima.apache.org

Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations, pages 1–4
Santa Fe, New Mexico, USA, August 20-26, 2018.

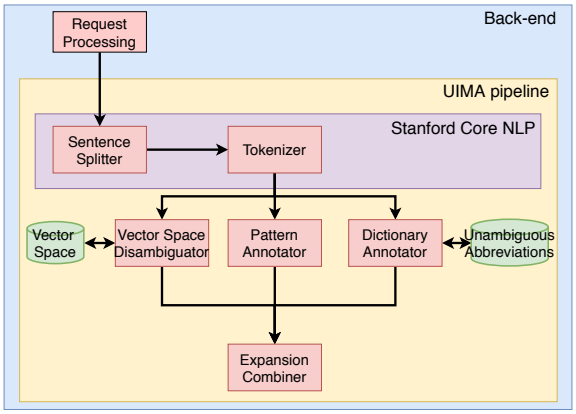

Figure 1: System architecture: input text is split and tokenized, defined abbreviations are extracted and expanded (Pattern Annotator), a dictionary resolves unambiguous cases (Dictionary Annotator), ambiguous cases are expanded using context (Vector Space Disambiguator); possible conflicts are resolved in the Expansion Combiner.

pipelines and benefits from a wide-array of external NLP libraries. The wide support for libraries allows us to employ established tools for pre-processing steps such as tokenization and sentence splitting for which we use the Stanford Core NLP library (Manning et al., 2014). We separate the abbreviation and acronym expansion into four different components: three components that perform expansion and one that combines their outputs in order to achieve consistency.

Before we describe the various components in detail, we establish some definitions. An *unambiguous abbreviation or acronym* is one that never expands into more than a single long-form. This corresponds to a one-to-one mapping. For example, in all of English Wikipedia we could only find one meaning for the acronym *SSRMS*, meaning *Space Station Remote Manipulator System*, popularly referred to as *Canadarm2*. *An ambiguous abbreviation or acronym* is one that can expand to multiple long-forms and the correct expansion is dependent on the context. However, *an unambiguous use* of an ambiguous abbreviation or acronym is one where an ambiguous abbreviation or acronym is used in such a way that the correct expansion is obvious. One such case is the definition of an abbreviation, such as *The Mobile Servicing System (MSS), is a robotic system on board the International Space Station*[3]. Because of the definition, it is clear which expansion is intended by the author.

The *Pattern Annotator* component is a re-implementation of the rules presented by Schwartz and Hearst (2003). It uses linguistic patterns to identify definitions of abbreviations and acronyms. More specifically, it looks for either the pattern *text (<short-form>)*, or the reverse *<short-form>(text)*. For each identified instance, it attempts to find a long-form expansion in the text preceding the parenthesis or contained in the parenthesis, respectively. This component can thus identify abbreviations and acronyms defined directly in the user-provided text. Our implementation differs from that of BADREX (Gooch, 2011) by the fact that it more closely follows the extraction rules defined in Schwartz and Hearst (2003). At the same time, we provide added support for various edge cases. For example, the original method does not support mapping of long-forms to short-forms when the long form contains two words at the beginning that start with the same letter (for example, *OAS* meaning *Organization of American States* is wrongly mapped to *of American States*). Our system solves this by a combination of looking ahead and a small set of stop-words not to be considered for the first word in a long-form (e.g. which, where, at, on, . . .).

The *Dictionary Annotator* component uses a dictionary of unambiguous abbreviations and acronyms, that we automatically extracted from English Wikipedia. We pre-processed English Wikipedia using only our Pattern Annotator in order to extract all abbreviations and acronyms that are unambiguous. Unambiguous abbreviations and acronyms have a one-to-one mapping between long-forms and short-forms. This annotator gives our system the ability to expand abbreviations and acronyms that are not

[3] https://en.wikipedia.org/wiki/Mobile_Servicing_System

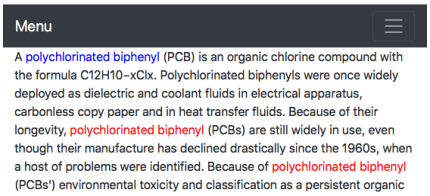

Figure 2: Screenshot of text with highlighted definition and added expansion of short forms to long forms

Figure 3: Screenshot of explanation information on long form source annotation that users can review.

defined in the text, but are known to only ever mean one thing. By focusing exclusively on unambiguous abbreviations and acronyms, this annotator avoids the pitfalls of dictionary based systems described in Section 3, i.e., the annotator avoids creating wrong expansions for abbreviations which can mean multiple things by working exclusively with abbreviations known to be unambiguous.

The third component that performs expansions, the *Vector Space Annotator* deals exclusively with ambiguous abbreviations and acronyms. It uses the context surrounding a short-form and a pre-computed vector space in order to disambiguate the abbreviation. The vector space is based on sentences from English Wikipedia containing ambiguous abbreviations (meaning abbreviations containing a one-to-many mapping between short-forms and long-forms) that are used in an unambiguous way (meaning that we already know which one of the multiple expansions is the correct one). We extracted these sentences using our *Pattern Annotator*. The *Vector Space Annotator* can thus expand abbreviations and acronyms that can have multiple meanings and whose definitions do not appear in the user provided text.

Finally, the *Expansion Combiner* uses the annotations from the previous components and combines them into consistent overall expansions. Please note that it is possible that two annotators expand an abbreviation to different long-forms. For example, the user provided text might introduce a new meaning for an abbreviation that we know as unambiguous and so, the *Pattern Annotator* and *Dictionary Annotator* might disagree. Similarly, the *Pattern Annotator* and *Vector Space Annotator* might arrive at different expansions if the author uses a new meaning for a known ambiguous abbreviation, or if the *Vector Space Annotator* should output an incorrect expansion. Finally, since the *Vector Space Annotator* works on one sentence at a time, it is possible that it disambiguates the same abbreviation to different long-forms in different sentences, thus leading to inconsistencies. The *Expansion Combiner* addresses these cases by implementing a priority system and, for the *Vector Space Annotator* specifically, a voting system.

Figure 2 shows a screenshot from the Abbreviation Expander system. In the example text, the original definition given in the text is marked and all subsequent uses of the short-form are preceded by the abbreviation's expansion. Users can verify how the system arrived at a specific expansion by clicking on the inserted expanded form, see Figure 3. The system features a menu where users can input or open some pre-loaded text files.

3 Related Work

BADREX (Gooch, 2011) is a plugin for the GATE (Cunningham et al., 2013) text analysis framework. It performs abbreviation expansion using dynamic regular expressions based on linguistic patterns for their definition (Schwartz and Hearst, 2003). The system requires an installation of GATE and familiarity with establishing GATE pipelines and loading plugins. It can identify abbreviation and acronym definitions in text and can then co-reference other instances of the identified short-forms to the found definition.

BADREX cannot perform abbreviation disambiguation, i.e., it cannot handle ambiguous cases as it relies exclusively on the definitions present in the document.

Web browser-based systems, like ABBREX (ABBREX, 2018), can be installed in a browser and expand abbreviations found on web pages. The expansion is based on stored dictionaries of abbreviations and lists of web pages they apply to. Thus, they cannot pick up definitions in text, or perform disambiguation. Being a dictionary-based expander, ABBREX assumes a one-to-one mapping between abbreviations and their long-forms, which means that in the case of ambiguous abbreviations, it has no other alternative, but to expand to whichever long-form is stored in the dictionary.

Another type of system, found e.g. in commercial software (Bartels Media, 2018; SmileOnMyMac, 2018), tries to expand user-defined abbreviations at writing time. This kind of software targets a different use case and cannot be applied to already written text.

The problem of matching abbreviations and acronyms with their long-forms has also been studied in research such as (Wu et al., 2015; Moon et al., 2015). However, they assume supervised learning settings, where a large amount of human effort has to go into providing ground truth examples. Also, they generally focus on methods, and do not provide (online) systems that users can easily use.

Abbreviation Expander presents a working web-based solution that does not require supervised ground truth information, and that can handle both unambiguous and ambiguous cases.

4 Conclusion

We present Abbreviation Expander, a web-based system that allows users to expand abbreviations and acronyms in text documents. Our system builds on state of the art methods for identification of abbreviations, acronyms and their definitions and our novel disambiguator based on word vector spaces. The *Vector Space Annotator* is still under active research and will be described in detail in a research paper in the near future. Abbreviation Expander requires no technical knowledge on part of its users and reliably expands both unambiguous and ambiguous abbreviations, improving text understanding and access in practice. In the future we plan to include feedback features into the system so that users can reject wrong expansions.

References

[ABBREX2018] ABBREX. 2018. ABBREX - The Abbreviation Expander. `http://abbrex.com`.

[Bartels Media2018] GmbH Bartels Media. 2018. WordExpander. `http://www.wordexpander.net`.

[Cunningham et al.2013] Hamish Cunningham, Valentin Tablan, Angus Roberts, and Kalina Bontcheva. 2013. Getting more out of biomedical documents with gate's full lifecycle open source text analytics. *PLoS computational biology*.

[Gooch2011] Phil Gooch. 2011. BADREX: In situ expansion and coreference of biomedical abbreviations using dynamic regular expressions. `https://github.s3.amazonaws.com/downloads/philgooch/BADREX-Biomedical-Abbreviation-Expander/Gooch_BADREX_biomedical_abbreviation_expansion_2012.pdf"`.

[Manning et al.2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

[Moon et al.2015] Sungrim Moon, Bridget McInnes, and Genevieve B Melton. 2015. Challenges and practical approaches with word sense disambiguation of acronyms and abbreviations in the clinical domain. *Healthcare inform. research*, 21(1):35–42.

[Schwartz and Hearst2003] A Schwartz and M Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. *Pacific Symp. Biocomp.*, 8.

[SmileOnMyMac2018] LLC SmileOnMyMac. 2018. TextExpander. `https://textexpander.com/`.

[Wu et al.2015] Yonghui Wu, Jun Xu, Yaoyun Zhang, and Hua Xu. 2015. Clinical abbreviation disambiguation using neural word embeddings. In *Proceedings of the 2015 Workshop on Biomedical Natural Language Processing (BioNLP)*, pages 171–176.