Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe. Alle Stellen, die aus den benutzten Quellen und Hilfsmitteln wörtlich oder inhaltlich übernommen sind, habe ich als solche kenntlich gemacht. Sofern KI-Tools verwendet wurden, habe ich Produktnamen, Hersteller, die jeweils verwendete Softwareversion und die Art der Nutzung (z.B. sprachliche Überprüfung und Verbesserung der Texte, systematische Recherche) benannt. Ich verantworte die Auswahl, die Übernahme und sämtliche Ergebnisse des von mir verwendeten KI-generierten Outputs vollumfänglich selbst. Die Satzung zur Sicherung guter wissenschaftlicher Praxis an der TU Berlin vom 8. März 2017 (https://www.static.tu.berlin/fileadmin/www/10000060/FSC/Promotion__Habilitation/Dokumente/Grundsaetze_gute_wissenschaftliche_Praxis_2017.pdf) habe ich zur Kenntnis genommen. Ich erkläre weiterhin, dass ich die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Berlin, den 26. April 2024

M. Reinhard

Unterschrift

Technische Universität Berlin
Fakultät IV: Elektrotechnik und Informatik
Institut für Telekommunikationssysteme
Fachgebiet Verteilte offene Systeme

Masters Thesis in Computer Science

Extending Semantic Hypergraphs by Neural Embedding-based Semantic Similarity for Pattern Matching

Max Reinhard
Matrikelnummer: 359417

26. April 2024

Supervised by Prof. Dr. Manfred Hauswirth
Additional guidance by Prof. Dr. Camille Roth,[*]
Dr. Telmo Menezes[*] and Dipl.-Math. Thilo Ernst[†]

[*] Centre Marc Bloch (An-Institut der Humboldt-Universität zu Berlin)
[†] Fraunhofer-Institut für offene Kommunikationssysteme

Abstract This thesis introduces a novel enhancement to the Semantic Hypergraph (SH) framework's pattern matching capabilities through the integration of Neural Embedding-based Semantic Similarity (NESS). The SH framework utilises a symbolic and structured representation of text, combined with a pattern matching functionality that allows for open and explainable text analysis – particularly aimed at Computational Social Science (CSS) researchers. Originally, the SH pattern matching follows strict, symbolic rules, with limited flexibility in matching the words that are integrated into the structured SH representation. Through incorporation of two NESS variants – Fixed Word Embedding and Contextual Embedding-based measures – the enhancement contributed in this work, allows for more semantically sophisticated patterns and therefore increases adaptivity of the pattern matching process. To provide partial control over the simultaneously introduced opaqueness, we enable users to configure crucial parameters of the NESS measurement. We conducted a comprehensive quantitative evaluation using a newly constructed labeled dataset to test the effectiveness of the NESS integration. The results demonstrate improved retrieval performance of the pattern matching, specifically showcasing the systems ability for contextual differentiation. Moreover, the implementation of the extended SH framework has been designed to be readily usable, facilitating its adoption for practical applications in CSS text analysis.

Zusammenfassung Diese Abschlussarbeit führt eine neuartige Verbesserung der Pattern-Matching-Fähigkeiten des Semantic Hypergraph (SH) Frameworks durch die Integration von Neural Embedding-basierter semantischer Ähnlichkeit (NESS) ein. Das SH-Framework nutzt eine symbolische und strukturierte Textrepräsentation, kombiniert mit einer Pattern-Matching-Funktionalität, die eine offene und erklärbare Textanalyse ermöglicht – insbesondere für Forscher im Bereich der Computational Social Science (CSS). Ursprünglich folgt das SH Pattern Matching strikten, symbolischen Regeln mit begrenzter Flexibilität bei der Übereinstimmung der Wörter, die in die strukturierte SH-Repräsenation integriert sind. Durch die Einbeziehung von zwei Varianten der NESS – statische Wort-Embedding und kontextsensitive Emedding-basiert — ermöglicht die in dieser Arbeit beigetragene Verbesserung komplexere semantische Patterns und erhöht somit die Anpassungsfähigkeit des Pattern-Matching-Prozesses. Um der gleichzeitig eingeführte Undurchsichtigkeit teilweise entgegenzuwirken, ermöglichen wir den Benutzern, entscheidende Parameter der NESS-Messung zu konfigurieren. Wir führten eine umfassende quantitative Auswertung mit einem neu erstellten, annotierten Datensatz durch, um die Effektivität der NESS-Integration zu testen. Die Ergebnisse zeigen eine verbesserte Retrieval-Leistung des Pattern Matching, insbesondere die Fähigkeit des Systems zur kontextuellen Differenzierung. Darüber hinaus wurde die Implementierung des erweiterten SH-Frameworks einfach zugänglich gestaltet, um eine praktische Anwendungen in der CSS-Textanalyse zu erleichtern.

Contents

1 Introduction

A significant part of the social world is nowadays being represented by digitally manifested text. Examples for this range from instant messaging, social media and any form of collective web activity to encyclopaedic websites, digitized libraries and government intelligence. The amount and richness of available social text makes it a valuable data source for social science research, while simultaneously creating an interest in automatic systems to analyze these texts on a large scale (Evans and Aceves 2016). Such research can be understood as part of the domain of *Computational Social Science* (CSS) (Lazer et al. 2009).

Systems based on techniques from the field of *Natural Language Processing* (NLP), as well well as the interlinked fields of *Information Retrieval* (IR) and *Information Extraction* (IE), have demonstrated great success in a variety of task related to text analysis. This success is largely attributed to the advancements of applying of *Machine Learning* (ML) and especially *Deep Learning* (DL) methods to text (Hirschberg and Manning 2015; Qiu et al. 2020). While being very effective at predicting or decision making, ML- and specifically DL-based systems generally do not deliver an explanation for their judgement and can mostly be viewed as "black box models" that are not transparent in their prediction or decision making process (Rudin 2019). Conversely this transparency and explainability is of high interest in CSS applications such as predicting political opinion based on social media activity (Wilkerson and Casas 2017).

The characteristic of *openness* – i.e. transparency and explainability – in text analysis systems typically conflicts with *adaptivity*, which refers to the capacity to learn and alter behaviour. This trade-off arises because adaptive systems frequently depend on ML or DL models, which are most often inherently *opaque*. In contrast, systems that are *open* are usually rule-based, making them principally *strict* in their operation.

The *Semantic Hypergraph* (SH) (Menezes and Roth 2021) is a framework for representing and analyzing *Natural Language* (NL). NL sentences can be modelled as an ordered, recursive hypergraph of hyperedges, which can be represented in a formal language. Based on this, the framework allows to specify semantic patterns in a pattern language, which can be matched against an existing SH. It aims to provide an *open* and *adaptive* system to analyse text corpora, especially in the domain of CSS. The framework is open in the sense that its representation formalism is inspectable and intelligible by humans and that the pattern matching follows explicit rules. It is adaptive in the sense that its parsing is built from ML-based subsystems and therefore allows for an error-tolerant tranlataion from NL to SH in regards to grammatical and syntactical correctness.

While the SH representation and its construction can be considered to fulfil the open-adaptive properties, the framework's pattern matching is purely symbolic and follows a set of fixed rules. It can therefore be considered to be *open-strict*. The SH pattern defined by a user may – among other things – specify the structure of the hyperedges that should match it and allows to describe different levels of generalisations for the structural matching. In the SH framework's original form, there is no possibility to define semantically sophisticated generalisations for the matching of hyperedge content, i.e. actual words that are contained within the text that is represented by an hyperedge.

By extending the the SH matching process with the capability to utilize some form of semantic similarity in regard to edge content, we allow the SH framework's users to define patterns, which describe such semantically sophisticated generalisations of edge content. There exists a great variety of approaches for determining the semantic similarity of text (Chandrasekaran and Mago 2021; Harispe et al. 2015). In this work, we identify *Neural Embedding-based Semantic Similarity* (NESS) measures to be best suited for an integration into the SH pattern matching. Since NESS measures are generally based on machine learning methods and sometimes based on deep neural network, they principally do not provide the openness that is inherent to the purely symbolic pattern matching process.

In the sense of the open-opaque / strict-adaptive classification, this integration means a shift from openness to opaqueness and from strictness to adaptivity. To counteract the opaqueness introduced by an NESS integration into the SH pattern matching, we allow user control over the generalisation level via the configuration of a similarity threshold, the ability to choose between fixed word and contextual neural embedding-based semantic similarity, and the option to toggle sub-edge specificity of the contextual embeddings. Therefore mantaining some openness while still benefiting from increased adaptivity.

In the context of this thesis, *Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching* (NESS-SHPM) is conceptualised and implemented by integrating it into the existing software package *graphbrain*[1]. Additionally it is subjected to a comprehensive quantitative evaluation regarding its retrieval performance on a newly constructed dataset comprised of labeled news headers.

The rest of this thesis is structured as follows: Chapter 2 introduces the necessary background in regards to the Semantic Hypergraph framework and discusses the state-of-the-art of semantic similarity measures. Chapter 3 outlines the problem introduced earlier in more detail and formulates the subsequent research questions. Chapter 4 devises the concept of NESS-SHPM as approach to solving the stated problem. Chapter 5 describes the specific software implementation of the proposed solution. Chapter 6 provides a thorough quantitive evaluation of the implemented system and discusses the results. Chapter 7 concludes this thesis by presenting the contributions made herein and suggesting possible avenues for further work.

[1] graphbrain: `https://graphbrain.net/`

2 Background

In this chapter the necessary background for this work is presented. This includes on the one hand the *Semantic Hypergraph* (SH) framework and its pattern matching capabilities in particular. On the other hand the concept of *Semantic Relatedness* (SR) and specifically *Semantic Similarity* (SS) is discussed with a focus on its realisation in the form of *Neural Embedding-based Semantic Similarity* (NESS).

2.1 Semantic Hypergraph Framework

The Semantic Hypergraph framework, introduced in Menezes and Roth 2021, offers a novel approach for the representation and analysis of NL text. At its core, the SH framework transforms text into a structured, symbolic format, thereby facilitating the analysis of text collections. A pattern language intrinsic to the SH framework enables the definition of patterns that encapsulate generalisations of text in the SH format.

Designed primarily for CSS researchers, the SH framework emphasises a high degree of openness, encompassing transparency and explainability. Additionally, the SH framework exhibits a level of adaptiveness, attributing to its reliance on ML components for the translation of NL text into the SH format. This adaptiveness reflects the framework's capability to evolve and accommodate the variable form of natural language.

One of the objectives of the SH framework is to bridge the existing gap in open-adaptive NLP systems. By providing a mechanism for structured symbolic representation and pattern-based analysis of text, the SH framework proposes a solution that combines the benefits of machine learning adaptiveness with the requisites of open and explainable research in CSS and beyond.

For a more comprehensive description of the Semantic Hypergraph framework, the original publication (Menezes and Roth 2021) should be consulted.

2.1.1 Structure and Syntax

In the SH framework, natural language is represented as a recursive, ordered *hypergraph* (in the following also referred to as *graph*). Such a hypergraph consists of *hyperedges* (in the following also referred to as *edges*). These hyperedges themselves can consist of a generally unrestricted number of hyperedges. To denote the hierarchy between hyperedges the terms *parent edge* and *child edge* or *sub edge* are introduced. If an edge e_A contains edges e_B and e_C, e_A is considered to be the parent edge of child or sub edges e_B and e_C. Since hyperedges are ordered, the edge $e_A = (e_B\ e_C)$ is different to an edge $(e_C\ e_B) \neq e_A$. If a hyperedge is not a composition of multiple hyperedges, it is called an *atom*.

Hyperedge Examples

Some example phrases and their corresponding hyperedges are given to help illustrate the general concept of the hypergraph representation and the hyperedge notation:

- "banana": banana/C, "eat": eat/C, "sweet": sweet/C
- "Mango is tasty.": (is/P.sc mango/C tasty/C)
- "Sophie throws the ball.": (throws/P.so sophie/C (the/M ball/C))

Hyperedge Components and Notation

An atomic hyperedge is noted as follows:

```
<CONTENT>/<TYPE> or
<CONTENT>/<TYPE>.<ARGUMENT ROLES>
```

The different edge components *content*, *type* and potentially *argument roles* are here represented by placeholders in brackets (<>). In an atom, the content is always noted in lowercase. Non-atomic hyperedges are noted as a composition of their sub edges, as illustrated above. They also always have content and type, which are not part of the notation, but can be inferred. Each of the hyperedge components is explained in the following.

Hyperedge Types

Every hyperedge in a semantic hypergraph has a specific type. All types that an edge can assume are listed in table 2.1, which is adapted from Menezes and Roth 2021, p. 7.

Code	Type	Purpose	Example
C	Concept	Define atomic concepts	**apple/C**
P	Predicate	Build relations	**(is/P** berlin/C nice/C)
M	Modifier	Modify a concept, predicate, modifier, trigger	**(red/M** shoes/C)
B	Builder	Build concepts from concepts	**(of/B** capital/C germany/C)
T	Trigger	Build specifications	**(in/T** 1994/C)
J	Conjunction	Define sequences of concepts or relations	**(and/J** meat/C potatoes/C)
R	Relation	Express facts, statements, questions, orders, ...	**(is/P berlin/C nice/C)**
S	Specifier	Relation specification (e.g., condition, time, ...)	**(in/T 1976/C)**

Table 2.1: Hyperedge types with their purpose and examples. For the examples the sub edge(s) that correspond(s) to the respective type is noted in bold face.

Type Inference Rules The type of a parent hyperedge is inferred based on the types and the order of its child hyperedges following specific inference rules. These rules can be inspected in Menezes and Roth 2021, p. 8, but are omitted here, because they are deemed mostly irrelevant for this work.

Hyperedge Content

The textual representation of an edge is referred to as *edge content* or *edge text*. For atoms this is generally a word, except for semantically relevant punctuation characters and special symbols (see below). Generally speaking, a hyperedges content is the reconstruction of the word, phrase or sentence that it represents, which can be produced by following the hyperedge structure.

Special Content Symbols The content of edges of the builder and conjunction type can also take on the form of special symbols +/B and :/J. This is utilised to represent phrases where there is no (meaningful) word or character attributable to the edges content. Examples of this are (taken from Menezes and Roth 2021, p. 7):

- "guitar player": (+/B guitar/C player/C)
- "Freud, the famous psychiatrist": (:/J freud/C (the/M (famous/M psychiatrist/C)))

Argument Roles

To make the meaning of the relation of sub edges in a parent edge more explicit, builder and predicate edges can have argument roles. In parent edges that start with an edge of one of these two types, the roles of the other edges can be specified by adding argument roles to the builder or predicate edges. These are denoted as shown in section 2.1.1. A table of all possible argument roles and an explanation of their functionality can be inspected in Menezes and Roth 2021, p. 8, but they are not in the focus this work.

2.1.2 Translation Process

For natural language text to be represented in the Semantic Hypergraph framework, it needs to be translated (or parsed). This aspect of the framework is mostly irrelevant for this work and is therefore only described briefly here. The translation is realised by machine learning based software components. This is a high-level description of the process:

1. Segment text into sentences and tokenise sentences
2. Annotate tokens with POS-tags, dependency labels and NER categories
3. Use a trained classification model to predict an atom edge type for a token using its annotation labels as input features
4. Recursively construct more complex hyperedges from the sequence of atomic edges by essentially applying the type inference rules

The edges that correspond to the sentences into which the original input text has been segmented are referred to as *root edges* of a graph in the following.

2.1.3 Pattern Matching

Text represented in the SH framework can be analysed via *Semantic Hypergraph Pattern Matching* (SHPM). Information can be extracted from texts represented as hypergraph by constructing patterns in the frameworks pattern language and matching these patterns against the graph. The extracted information can take on the form of e.g. specific kinds of expressions and parts of these expression that fulfil a certain role.

Pattern Language

The *Semantic Hypergraph Pattern Language* (SHPL) essentially allows for generalisations of hyperedge structure and content. All valid hyperedges are generally also valid patterns in the SHPL. Table 2.2 list all of the pattern languages elements (also referred to as operators), which are relevant to this work. Their functionality and therefore the form of generalisation that they enable is also described.

Element	**Notation**	**Function**
Variable	VAR	Matches any edge (restrictable by type and argument roles). Returns the edge that it captures.
Wildcard	*	Matches any edge (restrictable by type and argument roles).
Options list	[...]	Matches one of the given options. Applicable to edge type and content.
Innermost atom	>	Matches against the innermost atom of the edge that it captures. Removes an arbitrary level of nested modifiers.
Lemma function	lemma	Matches if the lemma of the captured edge content equals the given word. Combinable with an options list. Applicable only to atoms.
Argument role set	$\{\alpha\beta\gamma, \epsilon\}$	Allows for argument roles in arbitrary order. Comma-separated argument roles are optional

Table 2.2: Pattern language elements with their notation and function

Matching Process

At its core, the matching process involves comparing a predefined pattern against either an entire hypergraph, a set of edges, or a single edge. Independent of the matching target, this comparison is conducted one edge at a time, determining whether this edge aligns with the pattern. If pattern and edge do not match, the result of this process is only the binary information of *No Match*. If pattern and edge do match, the result of the matching process depends on whether the pattern contains variables. I this case, the result is not only the binary information of *Match*, but also the assignment of the variables contained in the pattern. It is possible for a single edge to yield multiple variable assignments, however this aspect will not be discussed further as it holds limited relevance for this work.

The matching process works recursively, processing both the structure and content of the edge in relation to the pattern. If a generalising part of the pattern (sub pattern) structurally matches a sub edge of the candidate edge, this sub edge is considered to be *captured* by this sub pattern. This is relevant for returning variable assignments and further processing, as is needed for the innermost atom operator or any content generalisation functionality.

Assuming the matching process is examining a specific sub pattern and sub edge, the content defined in the sub pattern is referred to as the *(pattern) reference content*. It is worth noting that a pattern may exclusively consist of structural elements, without specifying content. The content of the sub edge under comparison is termed as the *(edge) candidate content*. Apart of structural matching, the process also involves a comparison between the reference content and the candidate content, employing either a direct string-based comparison or comparing the candidate words lemma against the reference word(s).

Edge argument roles are ignored when matching, if they are not specified in a pattern. The matching of argument roles is not elaborated further, as it is of little relevance here.

Example Matchings To illustrate the expressiveness and of the SHPL and the functionalities of the different language elements, some example matchings of patterns against edges are shown in table 2.3. The corresponding edges and patterns are listed below.

Example Edges

$e_1 =$ (plays/P.so ann/C (acoustic/M guitar/C))

$e_2 =$ (plays/P.so ann/C (acoustic/M piano/C))

$e_3 =$ (plays/P.so ann/C (aggressive/M rugby/C))

$e_4 =$ (plays/P.sx ann/C (if/T (has/P.so she/C time/C)))

$e_5 =$ (play/P.so (and/J ann/C bob/C) (a/M song/C))

$e_6 =$ ((again/M perform/P.so) they/C (the/M (same/M song/C)))

Example Patterns

$p_1 =$ (plays/P PERS/C THING/C)

$p_2 =$ (plays/P ann/C *)

$p_3 =$ (plays/P ann/C VAR/[CS])

$p_4 =$ (plays/P ann/C >[guitar, piano]/C)

$p_5 =$ (plays/P ann/C >THING/C)

$p_6 =$ (plays/P ann/C (>IN*Similarity Threshold* (ST)/C [guitar, piano]))

$p_7 =$ ((lemma play/P) ACTOR/C THING/C)

$p_8 =$ ((lemma >[play, perform]/P) ACTOR/C THING/C)

Pattern	Edge	Match?	Variables
p_1	e_1	*Match*	PERS=ann/C, THING=(acoustic/M guitar/C)
p_1	e_4	*No Match*	
p_2	e_1, e_2, e_3, e_4	*Match*	
p_2	e_5, e_6	*No Match*	
p_3	e_1	*Match*	VAR=(acoustic/M guitar/C)
p_3	e_4	*Match*	VAR=if/T (has/P.so she/C time/C)
p_4	e_1, e_2	*Match*	
p_4	e_3, e_4	*No Match*	
p_5	e_1	*Match*	THING=guitar/C
p_5	e_3	*Match*	THING=rugby/C
p_5	e_4	*No Match*	
p_6	e_2	*Match*	INST=piano/C
p_6	e_3, e_4	*No Match*	
p_7	e_1	*Match*	ACTOR=ann/C, THING=(acoustic/C guitar/C)
p_7	e_5	*Match*	ACTOR=(and/J ann/C bob/C), THING=(the/M song/C)
p_7	e_6	*No Match*	
p_8	e_6	*Match*	ACTOR=they/C, THING=(the/M (same/M song/C)

Table 2.3: Example matchings of patterns against edges

2.2 Semantic Similarity Measures

Determining the *Semantic Relatedness* (SR) *Semantic Similarity* (SS) of two text items is a central task in the interlinked fields of natural language processing, information retrieval and information extraction. A text item can take on the form of a word, phrase, sentence paragraph or entire document. While SR and SS are often used synonymously, semantic relatedness can also be understood to entail a more nuanced description of the semantic relations of text items (Chandrasekaran and Mago 2021; Harispe et al. 2015). In this work, the term *Semantic Similarity* is therefore used to refer to the task of measuring the closeness, distance or similarity in regard to the meanings conveyed by two text items. This has to be differentiated from lexical (i.e. string-based) similarity measures, which in principle do not rely on the meaning of text items (P. and Shaji 2019).

There exists a great variety of approaches to measuring semantic similarity and their categorisation also takes on different forms in the literature. A structure that is in line with most of the reviewed literature employs the following categories: *Knowledge-based Semantic Similarity* (KBSS) (see section 2.2.1), *Corpus-based Semantic Similarity* (CBSS) (see section 2.2.2) and hybrid approaches which combine techniques from both categories (Chandrasekaran and Mago 2021; Han et al. 2021; Harispe et al. 2015; Zad et al. 2021).

The focus of this work in assessing the state-of-the-art of semantic similarity measures lies on CBSS – specifically a subform of corpus-based measures that is denoted here as *Neural Embedding-based Semantic Similarity* (NESS) (see section 4.2.3). These measures share some commonality with *Deep Learning-based Semantic Similarity* (DLSS), which is sometimes differentiated as an independent category in the literature (see section 2.3).

2.2.1 Knowledge-based Semantic Similarity

Knowledges-based semantic similarity measures rely on some form of underlying structured knowledge source to compute the similarity between two text items. These knowledge sources (referred to as knowledge systems in the following) may be ontologies, knowledge bases, knowledge graphs, semantic graphs, (lexical) databases and similar systems (Chandrasekaran and Mago 2021; Harispe et al. 2015). Examples of such structured knowledge systems are *ConceptNet* (Speer, Chin, and Havasi 2017), *WordNet* (Miller 1995), the *Paraphrase Database* (PPDB) (Ganitkevitch, Van Durme, and Callison-Burch 2013), *Wikidata* (Vrandečić and Krötzsch 2014), *DBpedia* (Auer et al. 2007) and *Yago* (Suchanek, Kasneci, and Weikum 2007).

Depending on the specific structure of the knowledge system, different methods of computing the semantic similarity are possible. Given two text items, they first need to be mapped to knowledge items – often called *concepts* – in the system. Then the similarity of these concepts can be calculated. There exists a variety of methods to calculate the similarity of two concepts, which can be categorised as follows: *structural or path-based*, *feature-based* and *information content (IC)-based*. Hybrid methods which combine multiple approaches also exists (Chandrasekaran and Mago 2021; Harispe et al. 2015).

Since most of the aforementioned knowledge systems generally have a graph-like structure, one type of measures is based on the spatial structure of the graph. The calculation of the similarity of two items is therefore mostly based on the length(s) of the path(s) between them. Another type of measure utilises the features that the two concepts exhibit, where the specific set of features is dependent on the employed knowledge system. A comparison of the two items feature values is the basis for calculating their similarity. IC-based methods rely on the "information provided by the concept when appearing in a context" (Sánchez and Batet 2013). This is closely related to the specificity of words measured by such metrics as *term frequency - inverse document frequency* (TF-IDF) (Aizawa 2003). Some of these methods determine the information content of concepts by using only information intrinsic to the knowledge system (or multiple knowledge systems). Other methods (additionally) utilize external information such as text corpora, so that the similarity measure could also be categorised as a hybrid of knowledge and corpus-based (see section 2.2.2). For a more in-depth exploration of knowledge-based similarity measures confer e.g. Chandrasekaran and Mago 2021; Harispe et al. 2015; Mihalcea, Corley, and Strapparava 2006; Zhu and Iglesias 2017.

2.2.2 Corpus-based Semantic Similarity

Corpus-based semantic similarity measures generally derive the semantics of text items and therefore their similarity not from a structured knowledge source, but from corpora of unstructured text. An assumption that is central to most of the approaches in this category is the *Distributional Hypothesis*, which states that words occurring in the same contexts tend to have similar meanings (Harris 1954). Consequently these approaches are also referred to as *Distributional Semantic Similarity Measures* (Mohammad and Hirst 2012b). Although not all corpus-based semantic similarity measures are based on the distributional hypothesis, it is considered to be the dominant and most relevant type of approaches in this category (Harispe et al. 2015, Section 2.4).

Vector Space Model

The Vector Space Model (VSM) (Turney and Pantel 2010) is the underlying framework for the text representation used by distributional semantic similarity measures. In this model, vectors represent text items, typically words, and are designed to encapsulate the distribution of words within a corpus. This generally involves counting word occurrences within particular contexts and organising these counts into a matrix. If the context is defined as a document (assuming the corpus is structured into distinct documents), the matrix is known as a *term-document* or *word-document frequency matrix*. Here, rows represent words and columns signify documents, with each cell indicating the frequency of a word's occurrence in a document. The context may also be defined more broadly as e.g. paragraphs, sentences, phrases, or a text window of arbitrary size (around another word). The entries in the resulting *word-context frequency matrix* can be weighted to reflect their specificity to the context, using measures such as TF-IDF or *(Positive) Point-wise Mutual Information* (Church and Hanks 1989; Niwa and Nitta 1994). Due to the potentially large vocabulary size and high number of context items, the word-context frequency matrix – and consequently the derived word vectors – may be high dimensional and sparse. It has been demonstrated that semantic similarity measurements benefit from techniques that smooth the matrix and therefore reduce the dimensionality of the vectors (Deerwester et al. 1990). A commonly employed method for this purpose is *Singular Value Decomposition* (SVD), or specifically its variant *truncated SVD*. Though other approaches, such as *Nonnegative Matrix Factorisation* (Lee and Seung 1999) or filtering of low entropy dimensions (Lund and Burgess 1996) have also been explored. A principle limitation of these approaches is, that they treat the context as a *bag of words* – ignoring word order beyond word proximity.

Vector Similarity Measures

In assessing the similarity between two vectors within the VSM, a variety of measures can be utilised to quantify their relationship. Spatial similarity measures treat the VSM as a geometric construct and determine a spatial relation of the vectors. The most notable examples include the Manhattan distance (L1 norm), the Euclidean distance (L2 norm), and cosine similarity. Alternative measures are mostly motivated by an information theory or probabilistic perspective on the VSM. Relevant examples among these are the Kullback-Leibler divergence and the Jensen-Shannon divergence (Mohammad and Hirst 2012b). It shall be noted, that a probabilistic interpretation entails constraints for the characteristics of the VSM. The choice of a suitable vector similarity measure is generally dependent upon the specific construction of the vectors in use. Cosine similarity, in particular, has been highlighted for its effectiveness (Mohammad and Hirst 2012a; Turney and Pantel 2010).

Spatial VSM-based Semantic Similarity Measures

This work focusses on spatially interpreted VSMs and their associated similarity measures, since they can be seen as a precursor to the neural embedding-based similarity measures, which are outlined and further elaborated on in section 2.3. In the following some distributional semantic similarity measures are presented, which were highly relevant for the development of the field. For each approach, we note the definition of the context for the word frequency matrix and the matrix' weighting and smoothing (i.e. dimensionality reduction) procedure.

Latent Semantic Analysis (LSA) (Deerwester et al. 1990; Landauer and Dumais 1997; Landauer, Foltz, and Laham 1998): In LSA, the context for the frequency matrix is a paragraphs. The matrix undergoes dimensionality reduction through a variant of Singular Value Decomposition (SVD), which reduces columns but retains rows, aiming to preserve the similarity structure among words. The similarity between word vectors, represented by their row values, is calculated using cosine similarity.

Hyperspace Analogue to Language (HAL) (Lund and Burgess 1996): HAL employs a sliding window to define its context, creating a word co-occurrence matrix with association strength values that depend on word proximity. Dimensionality reduction is achieved by excluding columns with low entropy. The vector similarity measure used is either Euclidean or Manhattan distance.

Correlated Occurrence Analogue to Lexical Semantics (COALS) (Rohde, Gonnerman, and Plaut 2006): COALS is a improved version of HAL and also uses a sliding window for its context. The matrix undergoes correlation normalisation and possibly SVD for smoothing and dimensionality reduction. Cosine similarity is used as vector similarity measure.

Explicit Semantic Analysis (ESA) (Gabrilovich, Markovitch, et al. 2007): ESA's frequency matrix context are Wikipedia articles, also called concepts. Weighting of the word-concept matrix is performed using TF-IDF. Semantic similarity is measured using cosine similarity between weighted concept vectors. The reliance on Wikipedia structure arguably renders ESA a hybrid combination of corpus- and knowledge-based approaches.

2.3 Neural Embedding-based Semantic Similarity

In natural language processing, the term *embedding* was first used in the context of Latent Semantic Analysis (Deerwester et al. 1990) to describe a vector representation of text. Since then, *text embedding* has become widely used in NLP to describe a dense, fixed-length vector representations of text items, which can be interpreted spatially to measure semantic relations between them (Almeida and Xexéo 2023).

Strictly speaking, *neural embeddings* of text refer to representations that have been produced by a neural network model. The development of these approaches is deeply interlinked with the advent of *Neural (Network) Language Modelling* (NNLM). The task of *language modelling* traditionally is to predict the most probable next word given a sequence of previous words (Chen and Goodman 1999). While Bengio, Ducharme, and Vincent 2000 are commonly referred to as the first to use a neural network for language modelling, Collobert and Weston 2008 are seen to be the first to build a NNLM with the focus on producing embeddings.

Neural Embedding-based Semantic Similarity (NESS) in this work refer to methods that compute the semantic similarity of text items based on a spatial measurement between their respective neural embeddings. NESS measures can be considered as a subform of CBSS, since they also fundamentally rely on the distributional hypothesis and employ the same general approach of deriving distributional semantics from a text corpus. Here, neural embedding-based approaches are further differentiated into *Fixed Word Neural Embedding-based Semantic Similarity* (FNESS) measures and *Contextual Neural Embedding-based Semantic Similarity* (CNESS) measures, which is elaborated in the following.

Fixed Word Embeddings vs. Contextual Embeddings

In this work, *(fixed) word embeddings* refer to neural text embeddings produced by models that are limited to single words. For a or given word, such a model will always generate the same embedding, hence the specification *fixed* (or *static*) word embedding. A significant consequence of this limitation is, that these word embedding models are not able to differentiate *homonyms*[1] or more specifically *homographs*[2] (Li and Yang 2018). These differentiations are only possible by taking the context in which the word appears into consideration. The context may be a phrase, sentence, paragraph or any sequence of tokens. A token can be word, but also the result of another kind of segmentation of the input text. While a sequence of words is generally processable using a fixed word embedding model, this results in a sequence of independently generated word embeddings. In this work, *contextual embedding* models are understood to incorporate the context. That means the generation of an embedding for a token contained in context is influenced by the other tokens in the sequence. This may result in different token embeddings for the same token given different contexts. Therefore contextual embeddings are principally able to represent different meanings of homographs and may also differentiate more subtle variations of word meaning, that can be derived from context (Liu, Kusner, and Blunsom 2020).

Deep Learning-based Semantic Similarity

Deep Learning-based Semantic Similarity (DLSS) measures are often treated as a distinct category in the literature, due to their substantially different underlying architectures and applicability to input text items (Chandrasekaran and Mago 2021; Han et al. 2021; Harispe et al. 2015; Zad et al. 2021). In this work, DLSS is not seen as a separate category, instead it is subsumed by the category of NESS. Deep learning-based approaches employ *Deep Neural Network* (DNN) models, which have multiple hidden layers, allowing them in principle to learn hierarchical of representations of the input data (Goodfellow, Bengio, and Courville 2016). In the context of NLP and SS specifically, DL models are most often designed to process text sequences, not individual words. Given a text item as input, the state of a specific hidden layer in the trained network after an inference pass is generally considered to be the embedding of the text item. From this perspective, all DLSS measures can be seen as neural embedding-based.

However, there also exists a variety of DLSS approaches, which use models that jointly process two input text items to directly predict their semantic similarity in the output layer. Here, the embeddings are only intermediate representations, but not explicitly used to measure spatial relations between them and they might not be suited for that. These methods are not considered to be NESS measures in our understanding of the term and are therefore not in the focus of this work. For such approaches confer e.g. He and Lin 2016; Lopez-Gazpio et al. 2019; Shao 2017; Tai, Socher, and Manning 2015; Tian et al. 2017; Tien et al. 2019a,b; Wang, Mi, and Ittycheriah 2017; Zheng et al. 2019.

Following the fixed word embeddings vs. contextual embedding distinction outlined above, DLSS approaches correlate strongly with the contextual embedding-based approaches, which is elaborated on in section 2.3.2.

[1] *homonym*: "A word that both sounds and is spelled the same as another word" (https://en.wiktionary.org/wiki/homonym

[2] *homograph*: "A word that is spelled the same as another word [...]" (https://en.wiktionary.org/wiki/homograph)

2.3.1 Fixed Word Embedding-based Semantic Similarity

Fixed Word Neural Embedding-based Semantic Similarity (FNESS) measures in this work denote semantic similarity measures that employ fixed word embedding models. Here, the concept of *neural embeddings* extends to encompass not only those embeddings directly derived from neural models, but also includes embedding generation approaches that incorporate elements of neural model methodologies and thereby achieving performances that rival strictly neural embeddings. This broader interpretation aligns with perspectives found in existing literature (Sezerer and Tekir 2021; Zuccon et al. 2015). Fixed word embedding models are generally based on non-deep machine learning methods and can be further divided into *prediction-based*, *count-based*, *hybrid* and *meta* models. In the following, some of the most relevant word embeddings models are presented.

Prediction-based Models

These mdoels are based on the concept of language modelling and work with local context windows. They are trained on predicting a target word (or token) given a context window of words (or tokens) or the other way around.

Word2Vec (Mikolov, Yih, and Zweig 2013; Mikolov et al. 2013a,b): Word2Vec has two forms; the *Continuous Bag of Words* (CBOW) model and the skip *Skip-Gram* (SG) model. CBOW predicts the target word given a context and SG predicts each context word given a target word. Both models are log-(bi)linear models, which basically learn a logistic regression with one hidden layer, whose weights are then used as embeddings. The best performing variant of Word2Vec is *Skip-Gram with Negative Sampling* (SGNS), which reportedly produced state-of-the-art results.

fasttext (Bojanowski et al. 2017): fasttext an extension of Word2Vec, which is reported to outperform SGNS. The central modification is, that the model works on subword tokens, which helps it to generalise to word which are unknown from the training data.

Count-based Models

These models are no neural embedding models in a strict sense, since they do not employ a neural network architecture. In terms of approach, they fall in line with spatially interpreted VSMs based on constructing a word-context frequency matrix (and are therefore fundamentally based on word counts) (Almeida and Xexéo 2023; Turney and Pantel 2010), which are discussed in section 2.2.2. In distinction to these approaches, the methods presented here also incorporate local context window information inspired by NNLM approaches and perform competitively when compared with strictly neural embeddings.

GloVe (Pennington, Socher, and Manning 2014) GloVe leverages corpus-wide, hence *global*, word co-ocurrences represented as word-word frequency matrix populated with probability ratios. The word co-occurrence probability ratios are weighted based on the distance of the word pairs in the context window from which they are extracted. It reportedly outperforms SGNS, given the same training corpus, training time and context window size.

LexVec (Salle, Idiart, and Villavicencio 2016; Salle and Villavicencio 2019; Salle, Villavicencio, and Idiart 2016): LexVec is based on the low-rank, weighted factorisation of the positive point-wise mutual information matrix via stochastic gradient descent. It incorporates local context window information by applying a weighting scheme to the matrix that has been shown to be implicitly performed by SGNS (Levy and Goldberg 2014). There also exists a subword based variant of LexVec, which is reported to perform competitively compared to fasttext (Salle and Villavicencio 2018).

Hybrid and Meta Models

Hybrid word embedding models in this work refer to approaches which are not only trained on unstructured text corpora, but also incorporate some form of structured knowledge source. This is akin to the ESA method presented in section 2.2.2. *Meta word embedding* models merge the information of multiple existing word embedding models into a new embedding model (Bollegala and O'Neill 2022). Both approaches are applicable simultaneously, resulting in a *hybrid meta word embedding* model.

Conceptnet Numberbatch (Speer, Chin, and Havasi 2017, 2018): Conceptnet Numberbatch is hybrid meta model that integrates structured knowledge from ConceptNet with existing word embedding models. First, ConceptNet-specific word embeddings are constructed by representing the knowledge graph's structure as a matrix of term-term connections, with cell values reflecting the sum of connecting edge weights. Further processing (i.a. smoothing and dimensionality reduction) is applied to this matrix and the resulting word embeddings are merged with Word2Vec, GloVe and fasttext embeddings. The method produced state-of-the-art-result, reportedly outperforming LexVec and all subsumed embedding models. A third-party comparative analysis of word embedding-based semantic similarity involving Word2Vec, fasttext, GloVe, LexVec and Conceptnet Numberbatch found, that the latter achieved best performance, when evaluating word similarity correlation with a ground truth (Toshevska, Stojanovska, and Kalajdjieski 2020).

MetaVec (García-Ferrero, Agerri, and Rigau 2021): MetaVec is in principle a pure meta word embedding model which integrates fasttext, ConceptNet Numberbatch, *JOINTChyb* (Goikoetxea, Soroa, and Agirre 2018) and *Paragram* (Wieting et al. 2015). However, JOINTChyb leverages data from WordNet, Paragram utilises the PPDB and Conceptnet Numberbatch relies on ConceptNet as described above. Consequently MetaVec is highly influenced by structured knowledge as well and can therefore also be considered to be a hybrid meta word embedding model. It is reported to outperform ConceptNet Numberbatch in multiple evaluations and in this work is seen to be the current state-of-the-art of fixed word embedding models for measuring semantic similarity.

2.3.2 Contextual Embedding-based Semantic Similarity Measures

Contextual Neural Embedding-based Semantic Similarity (CNESS) measures in this work denote semantic similarity measures, which employ contextual embeddings models. These models need to be able to jointly process sequences of tokens and produce the token embeddings interdependently (see section 2.3). Contextual embedding models are generally based on deep neural networks, which employ different architectural approaches to capture patterns in processing compositional data structures such as a token sequences (Goodfellow, Bengio, and Courville 2016). A wide array of DNN architectures has been applied to natural language processing and the generation of contextual embeddings specifically. The evolution of the current state-of-the-art of contextual embedding models is delineated here by focussing on three of the most relevant architecture types: *Convolutional Neural Networks* (CNN), *Recursive Neural Networks* (RNN) and *Transformer*-based models. Additionally some pivotal architecture-agnostic methodologies are introduced. Not all approaches presented here aim at producing embeddings in the spatially interpretable sense outlined above, but are still mentioned if they were highly influential on the development of the field. For a more comprehensive overview of deep learning based approaches to NLP, confer e.g. Min et al. 2023; Minaee et al. 2021; Young et al. 2018

Transfer Learning and Pre-training *Transfer learning* involves leveraging the representations learned from a different yet related task to train a model for a specific task (Zhuang et al. 2021). When these representations are derived from language modelling, the process is generally termed *pre-training* in NLP, notably advanced by Dai and Le 2015 for the generation of contextual embeddings.

Encoder-Decoder Design The *encoder-decoder* configuration is model design that can be applied to multiple DNN architecture types. It consists of two major model blocks: an *encoder* and a *decoder*. The encoder first processes the input data (e.g., token sequence in the source language) to generate a intermediate representation of fixed size, which is then passed to the decoder to generates the output data (e.g., translation in the target language). This model design has been proven very successful for sequence-to-sequence tasks such as machine translation and speech recognition, especially when the length of the input sequence and output sequence may differ (Cho et al. 2014).

Attention mechanisms Attention mechanisms are another important innovation in the processing of sequences, which help a model to focus on more relevant parts of the input sequence when producing an output sequence (Galassi, Lippi, and Torroni 2021). In successfully applying an attention mechanism to a machine translation task, the work of Bahdanau, Cho, and Bengio 2016 is a significant contribution in demonstrating the efficacy of this technique for NLP.

CNN and RNN Models

Convolutional neural networks extract patterns from compositional data structures through their convolutional layers or filters, adept at identifying spatial patterns, rendering them particularly effective for tasks such as image processing, but also natural language processing (Albawi, Mohammed, and Al-Zawi 2017). Recurrent Neural Networks are tailored for sequential data processing, with an architecture that allows preceding input elements to influence the processing of subsequent ones. This principally positions them well for handling token sequences – crucial in generating contextual embeddings (Yu et al. 2019).

Collobert and Weston 2008 are credited with being among the pioneers in employing CNNs for generating sentence embeddings. This approach has been further expanded by e.g. Kalchbrenner, Grefenstette, and Blunsom 2014. A significant contribution to the use of RNNs for language modelling was made by Mikolov et al. 2010, demonstrating promising results a in speech recognition task.

LSTM Models Within the spectrum of RNNs, *Long-Short Term Memory* (LSTM) models are probably the most relevant variant. These models are engineered to overcome the limitations RNNs face with long-term dependencies in input sequences (Hochreiter and Schmidhuber 1997; Yu et al. 2019). A notable development in the use of deep LSTM models for NLP was recorded in Graves, Mohamed, and Hinton 2013, where they reported state-of-the-art results in a speech recognition task.

In the following, some of the most relevant methods of producing contextual embedding by employing LSTMs in combination with other aforementioned techniques are presented.

context2vec (Melamud, Goldberger, and Dagan 2016): This method employs a bidirectional LSTM model to generate contextual embeddings, outperforming methods that rely on average pooling of word embeddings.

Context Vectors (CoVe) (McCann et al. 2018): CoVe leverages a bidirectional LSTM in combination with a form of attention mechanism to generate context embeddings by training on a machine translations task, thereby applying transfer learning. A concatenation of the CoVe model output and GloVe word embeddings is used as input for evaluation tasks, reportedly producing state-of-the-art-results.

Embeddings from Language Models (ELMo) (Peters et al. 2018): ELMo methods produces context-dependent embeddings based on pre-training a bidirectional language model implemented in form of a bidirectional LSTM. Similar to the CoVe, the ELMo embeddings are concatenated with word embeddings to form the input for downstream tasks, for which task-specific networks are augmented by convolutional or attention layers. This approach is reported to improve the state-of-the-art in six of the evaluation tasks.

Transformer-based Models

The *Transformer* architecture was first presented by (Vaswani et al. 2017), which was reported to significantly improve the state-of-the-art for a machine translation task. A key component is a specific form of the attention mechanism – called *self-attention* – that is used to weigh the importance of elements for each other when processing a sequence. Based in its application for machine translation, the original Transformer model follows the encoder-decoder design. Unlike recurrent networks, which process sequences element-by-element, transformer-based models handle entire sequences in parallel. This parallel processing capability facilitates pre-training these models on larger datasets and therefore effectively scaling them to larger model sizes, which is central to the performance gains on a multitude of NLP task, that are associated with their use (Min et al. 2023). In the following, some of the approaches most relevant for the evolution of Transformer-based contextual embeddings models are presented.

Bidirectional Encoder Representations from Text (BERT) (Devlin et al. 2019): BERT introduces a novel training objective in the form of masked language modelling. In this approach, certain tokens within an input sequence are obscured at random, with the goal of predicting these hidden tokens. Additionally, it is trained on a next-sentence prediction, which involves taking two separate sentences and determining whether the second sentence follows the first. Utilising Transformer encoders, BERT processes the input in a bi-directional manner during its pre-training phase. The method obtains new state-of-the-art results on eleven NLP tasks. Many variants of the approach have emerged, such as *RoBERTa* (Liu et al. 2019), which offers substantial performance enhancements over BERT through modification of the pre-training regime.

Embedding-specific Models While BERT and RoBERTa achieved new state-of-the-art performances on semantic similarity tasks, these models require that the input contains both sequences whose similarity should be determined. A similarity measurement is then produced by the output layer of the model, but the immediately generated embeddings are not well suited as input for spatial measures such as cosine similarity (Reimers and Gurevych 2019). This is hypothesised to be a consequence of anisotropic embedding spaces resulting from the masked language modelling objective (Li et al. 2020). Some of the most relevant approaches in resolving this limitation are presented in the following.

Sentence-BERT (SBERT) (Reimers and Gurevych 2019): SBERT utilises supervised fine-tuning on the foundational models BERT and RoBERTa through the use of siamese and triplet architectures. It leverages *Natural Language Inference* (NLI) datasets, which consist of sentence pairs categorised into entailment, contradiction, or neutrality towards one another, as a basis for training with classification, regression, and triplet loss objectives. This approach has demonstrated superior performance over existing methods in generating sentence embeddings, regarding all evaluated benchmarks for semantic similarity.

Simple Contrastive Sentence Embedding (SimCSE) (Gao, Yao, and Chen 2021): SimCSE builds on BERT and RoBERTa models, employing supervised and unsupervised techniques for sentence embedding. Utilizing an unsupervised strategy, SimCSE trains on sentence duplication using dropout to introduce noise. Its supervised method incorporates NLI data alongside a contrastive loss function, thereby producing state-of-the art results, surpassing SBERT in all evaluated semantic similarity benchmarks. Contrastive learning aims to learn representations by contrasting positive examples (similar or related data points) against negative examples (dissimilar or unrelated data points). The contrastive loss approach is hypothesised to create a more isotropic space for embeddings.

Massive Text Embedding Benchmark Since the field of contextual embeddings is rapidly evolving, it is not trivial to asses the current state-of-the-art and pinpoint the best performing model available for semantic similarity measurements. The *Massive Text Embedding Benchmark* (MTEB) (Muennighoff et al. 2023) offers support for this task. It is comprised of eight evaluation tasks, involving 58 datasets and 112 languages, including ten datasets for the semantic similarity task. An online leaderboard showcases the performance of all models that have been benchmarked by Muennighoff et al. 2023 and of those that have been submitted by third-parties.[3] In the following, two of the best performing models according to the online MTEB leaderboard at the time of access are presented.

EmbEddings from bidirEctional Encoder rEpresentations (E5) (Wang et al. 2022): E5 uses the same architecture as BERT and also initialises the model with the weights obtained from BERT's pre-training. This is followed by training using a contrastive learning approach with weak supervision signals derived from a large curated dataset of text pairs. The E5 model was developed in three variants: *small*, *base* and *large*, whose reported performance increases with size. There also exists a second version of the model (E5-v2), which was trained on a larger quantity and more diverse text pair datasets.[4] E5-v2 models

[3] MTEB Leaderboard: `https://huggingface.co/spaces/mteb/leaderboard` (Accessed on Oct. 9, 2023)

[4] See `https://huggingface.co/intfloat/e5-base-v2/discussions/1`

outperform the corresponding original models on most benchmarks, while performing on par in the semantic similarity task.

General Text Embeddings (GTE) (Li et al. 2023): Falling in line with with SimCSE and E5 in terms of approach, GTE also uses the BERT architecture and initialisation followed by contrastive learning process. Here it consists of a first step of contrastive pre-training with unlabelled data and a second step of contrastive fine-tuning with labelled task-specific data. GTE is also developed in three different model sizes – named *small*, *base* and *large* – which reportedly correlate with performance increases. Both, GTE and E5, construct the final sentence embedding by average pooling the produced token embeddings. Their publicly available implementation also share the same software interface. At the time of access, *GTE-large* is the top performing model regarding the average of all benchmarks in the MTEB and the average of the semantic similarity benchmarks specifically. It is therefore seen in this work to be the current state-of-the-art regarding contextual embeddings models for measuring semantic similarity.

3 Problem Statement

Computational Social Science (CSS) researches may typically be interested in extracting statements of a specific kind from a text corpus, such as expressions of sentiment of an actor towards some entity or expressions of conflicts between different actors. Two sensible ways to frame this task are as *text classification* (Kowsari et al. 2019) or *text retrieval* (Manning, Raghavan, and Schütze 2008). It can be addressed with a wide range of *automatic text analysis* systems. These systems are mostly based on techniques from the field of natural language processing, as well as the interlinked fields of information retrieval and information extraction (Chowdhary 2020).

A relevant perspective of categorising text analysis systems, especially from the point of view of CSS researchers, are the dimensions *open-opaque* and *adaptive-strict* (Menezes and Roth 2021). Here openness refers to the systems users ability to inspect and understand the processing, which we can also bee describes as transparency and explainability. These properties are of high interest in CSS applications such as predicting political opinion based on social media activity (Wilkerson and Casas 2017). An adaptive text analysis system does not (only) operate on strict rules, but is able to learn and modify its behaviour in some way. It is therefore in principle able to handle unforeseen variations in the text it processes. While both of these two properties are desirable for users are often found to be a trade-off. Current successful adaptive systems are most often based on machine learning models, especially deep neural networks (Hirschberg and Manning 2015), which are opaque in the way how they represent and process text (Rudin 2019).

The *Semantic Hypergraph* (SH) framework (see section 2.1) aims to fulfil both the open as well as the adaptive property of a text analysis system. It offers an inspectable and understandable representation of text that is constructed by a parser based on machine learning components. The SH representation and its construction can be therefore considered to fulfil the open-adaptive properties. The SH pattern matching language can be used to define patterns that match a specific subset of hyperedges in a given hypergraph. The matching process (described in section 2.1.3) is purely symbolic and follows a set of fixed rules. It can therefore be considered to be open-strict. In the context of the SH framework the CSS research task described above is better framed as text retrieval. The SH pattern acts as a *query* for which the most relevant items are retrieved. While the SH frameworks capabilities are not restricted to text retrieval, this work is focused on this application.

The SH pattern defined by a user may among other things specify the structure of the edges that should match it as well as their type (and the types of possible sub-edges). The SH pattern language allows it to describe different levels of generalisations for the structural matching. Additionally the actual words that should match need to be specified i.e. the reference content against which the edge content is matched, if the edge matches the pattern structurally. In the original form of the SH framework, the only way of generalising content matching is via the lemma functional pattern, but there is no possibility to define semantically more sophisticated generalisations.

This limitation in content matching generalisation capability means that all acceptable variants of edge content – i.e. specific words – must be explicitly provided by the SH framework's user. This also entails that a bias is introduced into the matching process due to the manual selection of words by the user defining a pattern. To better illustrate the problem, we consider the following sentences represented as hyperedges:

```
( plays/P ann/C piano/C )
```

Hyperedge 1: Represents the sentence "Ann plays piano"

```
( plays/P ann/C theatre/C )
```

Hyperedge 2: Represents the sentence "Ann plays theatre"

Hyperedge 1 and hyperedge 2 both follow the same structure, but differ in the content of the last sub-edge. Both edges are hence matched by pattern 1, which does not restrict content for this sub-edge. The SH pattern language also allows to define a pattern that matches both edges via a list of words as in pattern 2.

```
( plays/P ann/C */C )
```

Pattern 1: "Ann plays something" pattern

```
( plays/P ann/C [piano, theatre]/C )
```

Pattern 2: "Ann plays piano or theatre" pattern

Assuming we want to retrieve hyperedges that represent sentences which express something along the lines of "Ann engages in an artistic activity", we would probably also want to match hyperedge 3. However, to do so we need to extend the list of words in pattern 2 by "ballad" or resort to the more general pattern 1. Although this pattern also matches hyperedge 4, which is probably not what we aim for in meaning.

```
( plays/P ann/C ( a/M ballad/C ) )
```

Hyperedge 3: Represents the sentence "Ann plays a ballad"

```
( plays/P ann/C dead/C ) )
```

Hyperedge 4: Represents the sentence "Ann plays dead"

Utilizing some form of semantic similarity in regard to edge content in the SH matching process would allow the SH framework's users to define patterns, which describe generalisations of edge content. In the example outlined above, an option to include words that are semantically similar to "piano" and "theatre", would probably remove the need for the user to include all acceptable words that describe artistic activities in the pattern.

There exists a great variety of approaches for determining the semantic similarity of text as outlined in section 2.2. Fixed neural embedding-based semantic similarity measures – specifically hybrid models, which integrate structural knowledge in their principally corpus-based approach – have been identified in this work as one of the best performing methods to determine the similarity of single words (see section 2.3.1). An in integration of FNESS into the SH pattern matching could alleviate the problem of limited content generalisation as we have characterised it so far.

To retrieve all sentences that convey the meaning we are interested in, we probably also want the query pattern to match hyperedges that are more structurally diverse than those, which we have inspected above. Hyperedge 5 is such an example, which would be matched by the structurally more complex pattern 3. However, this pattern would also match hyperedge 6, which again is probably not what we want to express.

(plays/P ann/C (the/M (main/M character/C)) (in/T (a/M drama/C))))

Hyperedge 5: Represents the sentence "Ann plays the main character in a drama"

(plays/P ann/C */C */S)

Pattern 3: Structurally more complex "Ann plays something" pattern

(plays/P ann/C (an/M (important/M role/C) (for/T (her/M company/C)))

Hyperedge 6: Represents the sentence "Ann plays an important role for her company"

Given more complex hyperedge structures, it becomes increasingly difficult to construct patterns that may match all of the desired edges, especially if this set of edges is strongly defined by edge content. In this example we are interested in an interpretation of "play" that somehow relates to artistic activity. Since word semantics generally depend on textual context, so does the semantic similarity between them (Harispe et al. 2015, Section 2.2.3). An option to leverage some form of context sensitive semantic similarity for edge content in the SH pattern matching, would allow the user to express this form of context-specific content generalisations. From a user perspective this would involve providing content references in the form of larger text items such as sentences. Considering our example, this could be realised by defining the sentences represented by hyperedge 1, hyperedge 2, hyperedge 3 and hyperedge 5 as reference content.

Contextual semantic similarity measures generally aim to determine the SS between larger text items such as phrases, sentences or paragraphs (Verma and Muralikrishna 2020; Zad et al. 2021). However, the task of integrating contextual semantic similarity into the SH pattern matching is not solved by assessing the SS of entire sentences. Instead the content of specific sub-edges (the content of the predicate in our example) shall be semantically compared to the corresponding parts of the reference contents, thereby taking the context of of the sub-edge, i.e. the content of their root edge into account. Contextual neural embedding-based approaches allow this form of semantic similarity measurement since embeddings for parts of a larger text item can be constructed by combining the corresponding token embeddings. These sub-edge- or part-specific embeddings can then be used for SS measurement. Although DLSS measures directly operating on sentence level, may show better performance on established benchmarks (Chandrasekaran and Mago 2021), CNESS methods have been identified in this work to show strong performance as well in this regard (see section 2.3 and section 2.3.2).

Additionally, neural embedding-based models could also be integrated more efficiently into the SH framework's matching process than DLSS measures, which require the pairwise – computationally expensive – processing of text items to determine their similarity (Reimers and Gurevych 2019). In contrast, the embeddings of text items principally only need to be computed once and can then be used to perform the computationally inexpensive spatial measurements to determine their corresponding semantic similarities.

Integrating neural embedding-based semantic similarity measures into the pattern matching process would allow for edge content generalisation and therefore would make the process more adaptive. Since NESS measures are generally based on machine learning methods and CNESS measures specifically are mostly based on deep neural network, they principally do not provide the openness that is inherent to the purely symbolic pattern matching process of the SH framework. In the sense of the open-opaque / strict-adaptive classification described above, this integration would mean a shift from openness to opaqueness and from strictness to adaptivity. To counteract the opaqueness introduced by an NESS integration into the SH pattern matching, allowing user control over the generalisation level, can maintain some openness while still benefiting from increased adaptivity.

3.1 Research Questions

Based on the problem statement described previously, we articulate the following research questions. The *general research question* establishes a broader, more abstract research objective, while the *specific research questions* divide this overarching goal into smaller, more directly investigable queries.

3.1.1 General Research Question

R Can neural embedding-based semantic similarity measures be integrated into the pattern matching of the Semantic Hypergraph framework to allow for more semantically generalising matching regarding edge content while providing some control over the level of generalisation and therefore maintaining some openness of the pattern matching process?

3.1.2 Specific Research Questions

R.1 How can neural embedding based semantic similarity effectively and efficiently be integrated into the Semantic Hypergraph pattern matching?

R.2 Which specific neural embedding models would be the most suitable for assessing semantic similarity within the Semantic Hypergraph pattern matching process while addressing the challenges posed by contextuality?

R.3 How can the retrieval performance of the SH pattern matching be quantitatively evaluated?

R.4 Does integrating neural embedding-based semantic similarity measurements into the SH pattern matching improve the systems retrieval performance and how does it impact recall and precision?

R.5 Does the utilisation of contextual NESS enable the SH pattern matching to differentiate between desired and undesired edges in cases, which cannot be differentiated when utilising fixed word NESS?

R.6 How does measuring contextual NESS corresponding to sub-edge content influence retrieval performance in comparison to measuring the semantic similarity between the entire candidate and reference content context items?

R.7 How can the level of edge content related generalisation in the pattern matching process be effectively and transparently controlled and how does this impact precision, recall and general retrieval performance?

R.8 What impact does the the selection of a specific NESS model – both fixed word and contextual – have on SH pattern matching retrieval performance?

4 Solution Approach

The proposed solution to the research questions posed in section 3.1 is to conceptualise and implement the integration of neural embedding based semantic similarity into the pattern matching of the Semantic Hypergraph framework, followed by a suitable evaluation of this integration. This chapter outlines the concept of the approach, while the implementation and evaluation are detailed in chapter 5 and chapter 6 respectively.

The integration strategy involves pinpointing the most opportune point within the SH framework's pattern matching for the inclusion of NESS. This allows to identify which parts of the SH framework require modifications to accommodate NESS integration, as well as recognising any components that are currently missing. To facilitate this, the following section elaborates on the approach's general concept, addressing its core challenges and design decisions.

The system that is conceived by integrating NESS into the Semantic Hypergraph pattern matching will be referred to as *Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching* (NESS-SHPM)

4.1 Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching

The pattern matching process is described in section 2.1.3, where the concepts of *edge candidate content* and *pattern reference content* are introduced. At a certain point in the process, the candidate content is matched against the reference content. In the original form of the SH pattern matching, this is limited to exact string-based or lemma-based matching. The SH framework's functionality will be extended by neural embedding based semantic similarity measurement based matching regarding edge content.

The NESS measurement involves generating embeddings for the candidate and reference content, which in the following are referred to as *candidate embedding* and *reference embeddings*. The distance between these embeddings is calculated using a suitable metric and then compared to a given semantic *similarity threshold* t_s. If the measured semantic similarity exceeds this threshold, the contents of the pattern and the edge are considered to match. This process requires that the candidate content, reference content, similarity threshold (plus possibly additional other relevant parameters) are specified and passed to the component responsible for this matching step.

Moreover, the integration of NESS-based content matching into the SH pattern matching process will take two forms: *Fixed Word Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching* (FNESS-SHPM) and *Contextual Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching* (CNESS-SHPM). The particularities of these two system variants are elaborated in the following.

In the case of FNESS, embeddings are generated based on single words. Therefore the candidate content takes on the form of a *candidate word* while the reference content may be one or multiple *reference words*. It is considered potentially useful to leverage the generalisation capabilities that already exist in the SH pattern matching in the form of the lemma function. Hence a lemma based variant of FNESS is also conceptualised in the form of *Lemma-based Fixed Word Neural Embedding-based Semantic Similarity* (LFNESS). In the case of *Lemma-based Fixed Word Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching* (LFNESS-SHPM), the candidate content takes on the form of the candidate words lemma.

In the case of CNESS, embeddings are generated based on context which can be phrases or entire sentences. In the SH framework those are represented in the form of hyperedges and therefore the candidate and reference content take on the form of a *candidate edge* and one or multiple *reference edges*.

4.2 Integration into the Semantic Hypergraph Framework

The Semantic Hypergraph framework offers two primary avenues for enhancement: through the expansion of its pattern language or through the adaptation or refinement of the software interface utilised in the matching process. These approaches are not mutually exclusive and can be effectively combined.

Leveraging the core principles of the SH framework, the most logical and straightforward method to enrich its pattern matching capabilities is by extending its pattern language. This strategy also facilitates the concurrent application of NESS-based content matching alongside string- or lemma-based content matching within the same matching process.

Therefore, the incorporation of NESS-SHPM into the existing SH framework encompasses three principal components: the extension of the pattern language, the requisite modifications to the pattern language processing and adding the components necessary for the NESS measurement calculations.

A visual representation of the solution approach is depicted by fig. 4.1.[1] In this figure two different candidate edges are matched against two patterns, which make use of the extended pattern language. One of the illustrated matching processes utilises FNESS-SHPM for the second concept sub-edge (which is the object to the predicate sub-edge), while the other leverages CNESS-SHPM directly for the predicate sub-edge.

4.2.1 Extention of the Pattern Language

For the inclusion of NESS-based content matching within the SH pattern language, the *SemSim* functional pattern will be introduced. This addition is targeted at the functional pattern segment of the language, which is already predisposed to support flexible enhancements for SH pattern matching. This approach is consistent with the existing mechanism of realising lemma-based content matching generalisation capabilities in the framework.

[1] This diagram utilises the pattern `( (semsim-ctx */P ) ann/C */C * )`. It shall be noted, that the last wildcard operator * is not restricted by hyperedge type or argument roles and is therefore meant to match any sub-edge. This also includes no sub-edge, i.e. the pattern matches all reference edges.

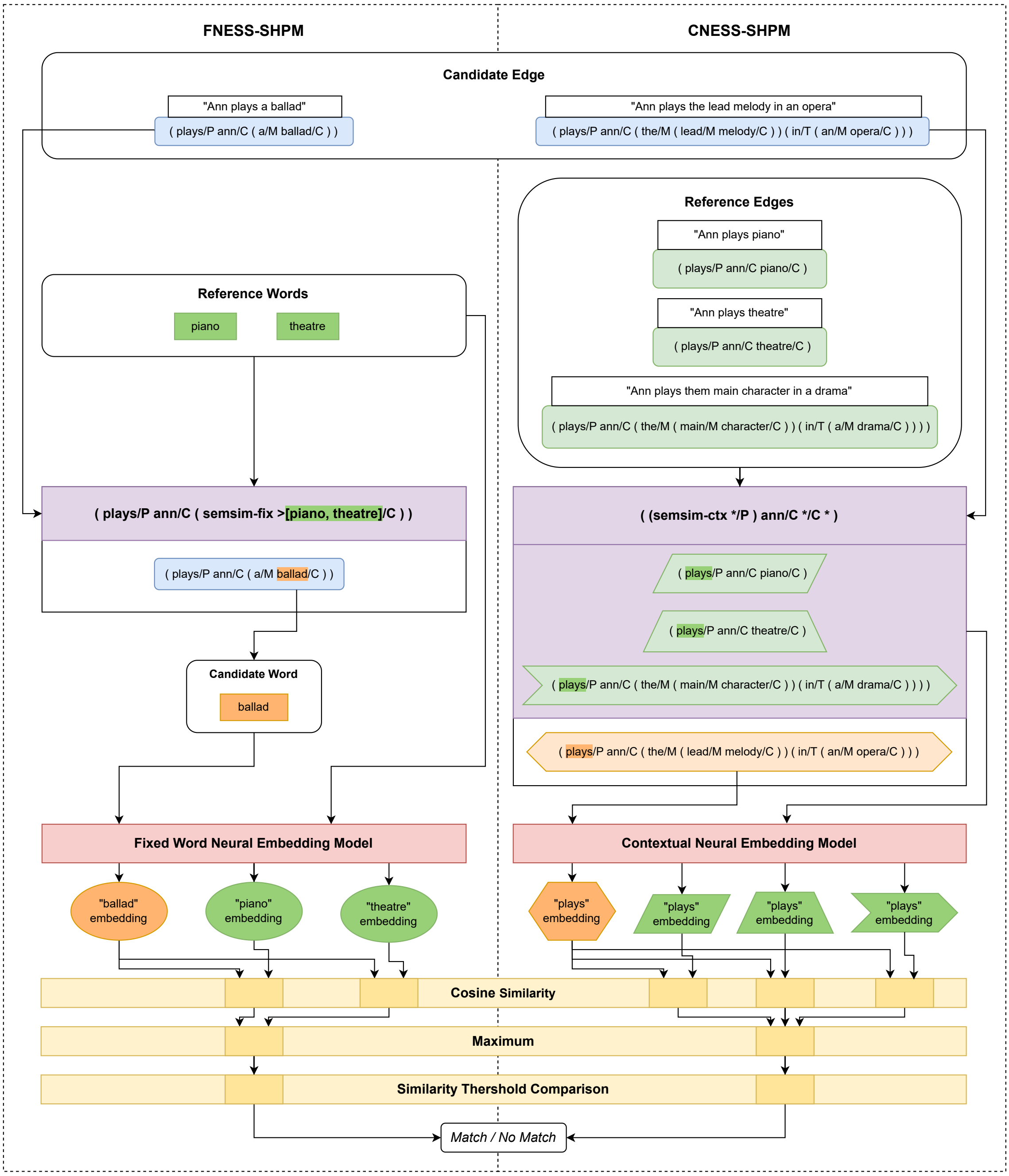

FNESS-SHPM
CNESS-SHPM
Candidate Edge
"Ann plays a ballad"
(plays/P ann/C (a/M ballad/C))
"Ann plays the lead melody in an opera"
(plays/P ann/C (the/M (lead/M melody/C)) (in/T (an/M opera/C)))
Reference Edges
"Ann plays piano"
(plays/P ann/C piano/C)
"Ann plays theatre"
(plays/P ann/C theatre/C)
"Ann plays them main character in a drama"
(plays/P ann/C (the/M (main/M character/C)) (in/T (a/M drama/C))))
Reference Words
piano
theatre
(plays/P ann/C (semsim-fix >[piano, theatre]/C))
(plays/P ann/C (a/M ballad/C))
((semsim-ctx */P) ann/C */C *)
(plays/P ann/C piano/C)
(plays/P ann/C theatre/C)
(plays/P ann/C (the/M (main/M character/C)) (in/T (a/M drama/C))))
(plays/P ann/C (the/M (lead/M melody/C)) (in/T (an/M opera/C)))
Candidate Word
ballad
Fixed Word Neural Embedding Model
Contextual Neural Embedding Model
"ballad" embedding
"piano" embedding
"theatre" embedding
"plays" embedding
"plays" embedding
"plays" embedding
"plays" embedding
Cosine Similarity
Maximum
Similarity Thershold Comparison
Match / No Match

Figure 4.1: Visualisation of the structure of both NESS-SHPM variants, each matching a candidate edge against a pattern. Left side depicts FNESS-SHPM, while the right side illustrates CNESS-SHPM.

SemSim Functional Pattern

When constructing a SH pattern that should be matched against a hypergraph, the *SemSim* functional pattern can be utilised at those points in the pattern, where content should be matched by NESS. The added SemSim functional pattern – also referred to as SemSim pattern – has one of two syntactical structures: One that does include a variable and one that does not. Both variants of the SemSim pattern and examples of their usage are illustrated in the table 4.1. The SemSim pattern consists of different syntactical components, represented by placeholders that are enclosed in brackets (<>). These components are explained in detail below. Examples usages of SemSim patterns as sub-pattern inside other patterns are shown below the syntax component explanations.

No Variable	<SF>	<IA>	<RC>	/<HT>.<AR>		<ST>
	semsim-fix		piano	/ C		
	semsim-fix	>	piano	/ C		
	semsim-fix		piano	/ C . so		
	semsim-fix		[piano, theatre]	/ C		
	semsim-fix		piano	/ C		0.5
	semsim-fix	>	[piano, theatre]	/ C . so		0.5
	semsim-fix-lemma	>	[piano, theatre]	/ C . so		0.5
	semsim-ctx		*	/ C . so		0.5
With Variable	<SF>	<IA>	<VA>	/<HT>.<AR>	<RC>	<ST>
	semsim-fix	>	ART	/ C . so	[piano, theatre]	0.5
	semsim-fix-lemma	>	ART	/ C . so	[piano, theatre]	0.5
	semsim-ctx		ART	/ C . so	*	0.5

Table 4.1: Syntactical structure of SemSim Pattern with specific examples.
Syntactical components: SemSim Function (SF), Reference Content (RC), Variable Declaration (VA), Hyperedge Type (HT), Argument Roles (AR), Innermost Atom Operator (IA), Similarity Threshold (ST)

SemSim Function (SF) The semsim function can be one of the following: `semsim-fix`, `semsim-fix-lemma` and `semsim-ctx`. These functions correspond to fixed word neural embedding-based, lemma-based fixed word neural embedding-based and contextual neural embedding-based semantic similarity measurement.

Reference Content (RC) The reference content is used to specify the reference word(s) for the FNESS variant. The square bracket notation which is already part of the SH pattern language is leveraged to pass multiple words as a list. The reference edge(s) that is needed for the contextual NESS variant is not given via the semsim functional pattern itself, but as a software interface argument to the matching process due to practical considerations (see section 4.2.1).

Variable Declaration (VA) If the content of the sub-edge captured by the SemSim sub-pattern should be captured in a variable, the second variant of the syntax applies. This notation is in line with the usage of the lemma functional pattern illustrated in section 2.1.3.

Hyperedge Type (HT), Argument Roles (AR) and Innermost Atom Operator (IA) These pattern components can be utilised in the same way as illustrated above in section 2.1.3 for the lemma functional pattern. Hyperedge type and argument roles are matched following the symbolic rules also described above. The innermost atom operator (>) can be placed in front of the reference content or the variable declaration if applicable.

Similarity Threshold (ST) The similarity threshold t_s can optionally be specified for a specific occurrence of the SemSim functional pattern, but it can also be given as a global parameter of the matching process (see section 5.3.1).

Example Usages

To illustrate the usage of the SemSim functional pattern as sub-pattern in another pattern, some examples are given below. We restrict these examples here to the `semsim-fix` pattern, since the `semsim-ctx` pattern additionally requires the definition of reference edges.

(plays/P ann/C (semsim-fix piano/C))

Pattern 4: "Ann plays something similar to piano" pattern

(plays/P ann/C (semsim-fix [piano, theatre]/C))

Pattern 5: "Ann plays something similar to piano or theatre" pattern

(plays/P ann/C (semsim-fix theatre/C 0.5))

Pattern 6: "Ann plays something similar to theatre" pattern with $t_s = 0.5$

Limitations of the Pattern Language Extension

The utilisation of the fixed variant of NESS-SHPM is possible solely via the SemSim functional pattern. This means that all information that is required to apply FNESS-based content matching, specifically the reference content and the similarity threshold, can be included in a SH pattern. In contrast, it was found impractical to provide the reference edges needed for the contextual NESS variant via a SH pattern. While the reference content argument could generally be used for that, this would result in very exhaustive patterns and impair their readability for humans. Therefore the conceptual design choice was made to pass the reference edges via the software interface of the SH pattern matching.

4.2.2 Modification of the Pattern Language Processing

This section outlines the integration of SemSim functional pattern processing with the current pattern language processing. The process is triggered when a pattern, incorporating a SemSim pattern, is matched against a specific hyperedge and the SemSim pattern is reached without any prior mismatches. At this point, the SemSim pattern captures a subedge for processing, which includes the symbolic structural matching. Additionally the necessary information for NESS measurement computations is extracted.

The operation of SemSim pattern processing changes based on the specific SemSim function applied. With FNESS-SHPM or LFNESS-SHPM, the process extracts the candidate word(s) or their lemma(s) from the identified sub-edge and the reference word(s) from the pattern. This extraction process requires the sub-edge to be atomic, which implies combination with the innermost atom operator to ensure functionality in all cases. For CNESS, it conducts a standard string-based matching of the reference content. Therefore utilizing the wildcard operator as a reference content argument is sensible in most practical use cases. For any NESS-SHPM variant, the similarity threshold is extracted from the SemSim pattern, if given.

Existing components manage the symbolic matching of hyperedge type and argument roles, as well as the application of the innermost atom operator, consistent with the standard pattern matching. The SemSim function, along with the similarity threshold (when provided), candidate words and reference words for (L)FNESS-SHPM, are passed to the newly designed NESS computation components.

4.2.3 Neural Embedding-based Semantic Similarity Measurement

Upon reaching this phase, the NESS-SHPM process has already either extracted candidate and reference contents via pattern language processing (in case of FNESS-SHPM) or received them through software interface arguments (in case of CNESS-SHPM). Similarly, the similarity threshold has been either derived from the SemSim pattern or assigned as a default parameter through the software interface. Additionally the *NESS type* (i.e FNESS or CNESS) is determined by the SemSim function and a specific *NESS model* (i.e. neural embedding model) has been specified through the software interface. Consequently, the process is set to obtain the respective embeddings for the candidate and reference content, which are central for conducting the semantic similarity assessment.

We denote the arguments of the NESS measurement (A_{NESS}) as follows: Pattern p, candidate content c_{can}, reference content(s) C_{ref} and similarity threshold t_s. The candidate content is either a candidate word w_{can} or a candidate edge e_{can}. The reference content is either a set of reference words W_{ref} or a set of reference edges E_{ref}. The similarity threshold is either extracted from the pattern $t_{s_{\text{pat}}}$ or given as a default parameter $t_{s_{\text{def}}}$.

$$A_{\text{NESS}} = \{p, c_{\text{can}}, C_{\text{ref}}, t_s\}$$

$$c_{\text{can}} \in \{w_{\text{can}}, e_{\text{can}}\} \ , \ C_{\text{ref}} \in \{W_{\text{ref}}, E_{\text{ref}}\} \ , \ W_{\text{ref}} = \{w_0, \dots, w_n\}, E_{\text{ref}} = \{e_0, \dots, e_m\} \ , \ n, m \in \mathbb{N}^+$$

$$C_{\text{ref}} = \begin{cases} W_{\text{ref}} & \text{if } c_{\text{can}} = w_{\text{can}}, \\ E_{\text{ref}} & \text{if } c_{\text{can}} = e_{\text{can}}. \end{cases}$$

$$t_s \in \{t_{s_{\text{pat}}}, t_{s_{\text{def}}}\}$$

Fixed Neural Embeddings

The creation of fixed neural embeddings necessitates only the candidate and reference words. Given that each word maps directly to a specific embedding, a straightforward lookup in the chosen fixed neural embedding model suffices to produce the necessary embeddings for both candidate and reference. The embedding v_w of word w, generated by fixed word neural embedding model M_{fix} is denoted as:

$$v_w = M_{\text{fix}}(w),\ V_W = \{v_w \mid w \in W\}$$

Contextual Neural Embeddings

The construction of contextual embeddings for the candidate and reference content involves a sequence of steps, premised on the assumption that these embeddings should mirror the specific sub-edge of the candidate edge captured by the SemSim pattern and its analogous sub-edge(s) in the reference edge(s). Thereby the reference edges need to structurally match (with disregard to edge content) the pattern, which contains the SemSim functional pattern, so that these reference sub-edges can be identified.

An alternative approach to constructing contextual neural embeddings does not account for sub-edge matching and only consists of the initial two stages of the construction process outlined below. This variant of the contextual embedding construction is in the following referred to as the *All Tokens* (AT) option.

1. Reconstructing phrases or sentences from the candidate and reference edges (deemed context items).
2. Producing embeddings for these context items using the designated contextual neural embedding model.
3. Identifying the sub-edge within the reference that aligns with the candidate's sub-edge captured by the SemSim pattern.
4. Associating the identified candidate and reference sub-edges with their respective sub-embeddings derived from the context item embeddings.
5. Generating the final embeddings for both candidate and reference based on these sub-embeddings.

The embedding v_e of edge e, generated by the contextual neural embedding model M_{ctx} when considering pattern p, is denoted as:

$$v_e = M_{\text{ctx}}(e, p),\ V_E = \{v_e \mid e \in E\}$$

Embedding Similarity Measurement

The procedure calculates the cosine similarity between the embeddings of candidate and reference content, comparing it to the pre-established similarity threshold. If the resultant similarity score surpasses this threshold, the candidate and reference contents are considered to match. Given multiple reference content items (and therefore multiple reference embeddings), the pairwise cosine similarities between the candidate and the reference embedding is computed. The maximum of these similarities is then compared to the similarity threshold to asses whether the candidate and reference contents match. Semantic similarity f_{semsim} given the arguments $A_{\text{NESS}} = \{p, c_{\text{can}}, C_{\text{ref}}, t_s\}$ is therefore calculated by:

$$f_{\text{semsim}}(A) = max(\{cos_{\text{sim}}(v_{\text{can}}, v_{\text{ref}}) \mid v_{\text{ref}} \in V_{\text{ref}}\}) \geq t_s$$

$$V_{\text{ref}} = \begin{cases} V_{W_{\text{ref}}} & \text{if } v_{\text{can}} = v_{w_{\text{can}}}, \\ V_{E_{\text{ref}}} & \text{if } v_{\text{can}} = v_{e_{\text{can}}}. \end{cases}$$

Cosine similarity is chosen as spatial measure, since it has been show well suited for assessing semantic similarity in vector spaces (see section 2.2.2) and is an established measure in the context of neural text embeddings (see section 2.3). The maximum of the pairwise similarities is used for comparison with the similarity threshold, because in case of FNESS, we want the reference words always to be deemed semantically similar to themselves. This would not necessarily be the case when using e.g. the mean of the pairwise similarities. However, other methods such as e.g a weighted mean could principally also be of interest, especially for CNESS.

5 Implementation

In this chapter we will present and explain the implementation of the solution approach conceived in chapter 4, which takes the form of a software realisation of *Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching* (NESS-SHPM).

The entailing software design and development is centred around integrating the necessary functionality in the existing publicly available software package *graphbrain*[1], which includes an implementation of the Semantic Hypergraph framework. This integration consists of two primary aspects: Modifying and augmenting the existing pattern language processing components to accommodate the extended pattern language (confer section 4.2.1 and section 4.2.2), as well as constructing new software components to realise the NESS measurement (confer section 4.2.3).

Implementation details, which are considered not to contribute to the general understanding of the software system, are omitted or have been modified for better readability and clarity. This means that all code listing presented below are pseudocode.

The particular version of graphbrain employed in this work is available in a dedicated repository, established specifically for this purpose.[2]

5.1 Overview of *graphbrain*

This section offers a concise overview of the essential elements of the graphbrain package, detailing its software architecture. It also describes specific aspects of the hypergraph data structure and the NL to SH parsing procedure that are of peripheral relevance here.

The graphbrain package contains the `graphbrain` Python module, which consists of several submodules. The architecture of the key parts of the modified and extened `graphbrain` module is depicted in fig. 5.1. This diagram highlights the primary call chain and data flow involved in the pattern matching process, while the return chain is excluded to enhance the diagram's clarity. Further details on the software structure and procedural flow are elaborated in subsequent sections.

Here, we focus on two particular submodules: the `graphbrain.patterns` submodule, which is primarily responsible for the pattern matching process and has therefore been significantly modified and expanded, and the newly created `graphbrain.semsim` submodule, which hosts the implementations of NESS measurements.

Of secondary importance are the `graphbrain.hypergraph` and `graphbrain.hyperedge` submodules, which are responsible for implementing the structures of the hypergraph and hyperedge, respectively.

[1] graphbrain: `https://graphbrain.net/`

[2] Thesis implementation: `https://github.com/wuxmax/graphbrain-thesis`

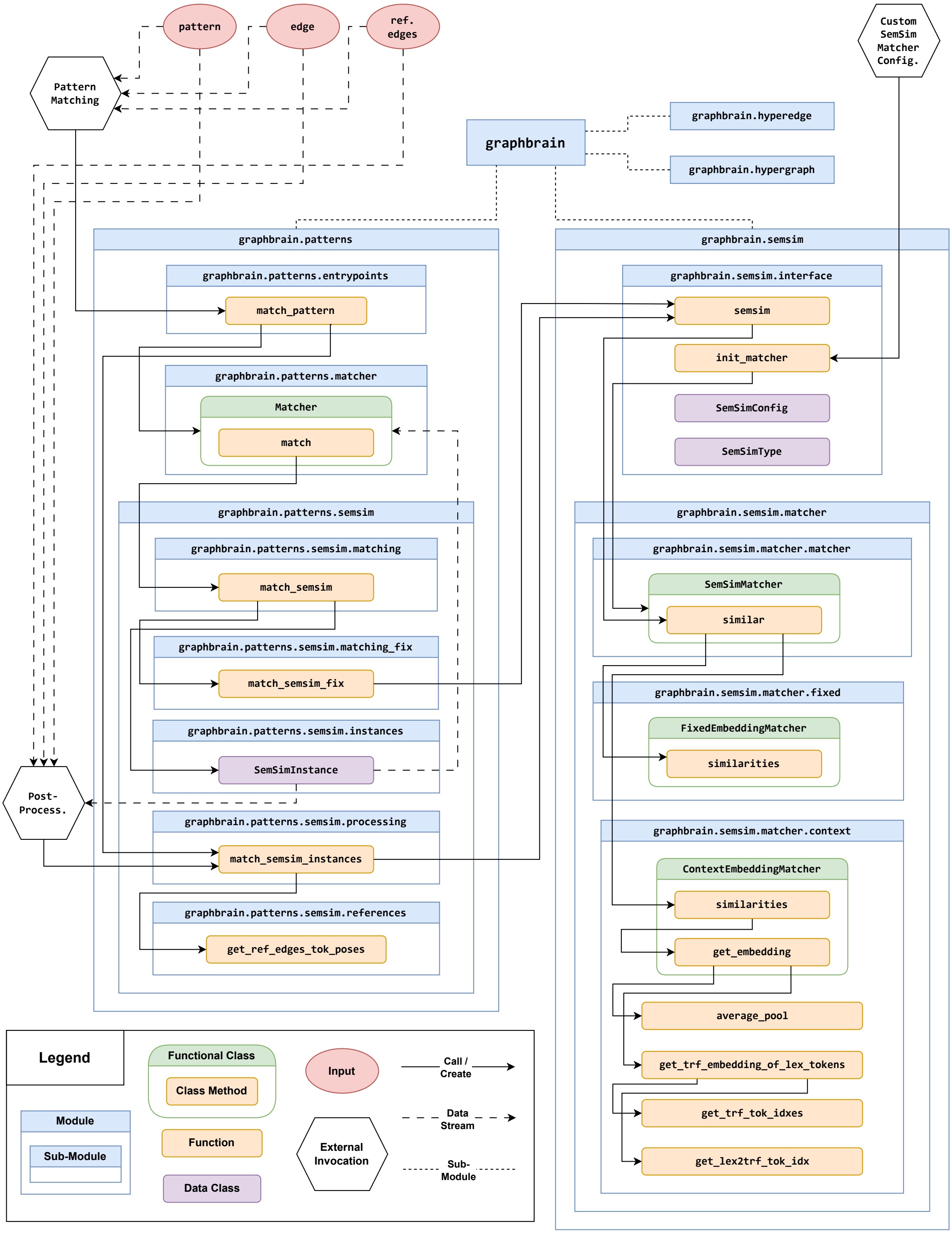


Figure 5.1: Simplified architecture diagram of the modified and extended `graphbrain` Python module, with primary call chain and data stream for the pattern matching process (return chain is omitted for visual clarity).

Hypergraph Implementation

A specific hypergraph is implemented in form of a key-value storage, for which currently *LevelDB*[3] and *SQLite*[4] are support as database backends. Interfaces for the key-value storages are implemented in `graphbrain.memory`. However, details of this implementation are irrelevant for this work.

NL to SH Parser Implementation

The translation process from natural language to the Semantic Hypergraph representations (see section 2.1.2), depends significantly on the established NLP library *spaCy* (Honnibal et al. 2020). Although the translation itself holds limited relevance for this work, it is important to note that the atomic edges in a SH representation of a sentence closely relate to the tokens resulting from the tokenisation of this sentence by spaCy's lexical tokeniser.[5]

5.2 Pattern Processing: `graphbrain.patterns`

The `patterns` module has been restructured from a monolithic design into multiple submodules to facilitate implementing the necessary modifications to the pattern language processing (see section 4.2.2), which are required to support the extended pattern language (see section 4.2.1). This includes the creation of the `patterns.semsim` submodule. Furthermore, some adjustments to the software architecture have been made to simplify the implementation of contextual NESS extended pattern matching (see section 5.2.1). Apart of the SemSim-specific submodule, the `patterns.entrypoints` and `patterns.matcher` submodules are also of relevance for this work.

Some general intricacies of the implementation NESS extended Semantic Hypergraph pattern matching are discussed below, before the specific implementation in the aforementioned submodules is presented.

In-Line- vs. Post-Processing

While the processing of the SemSim functional pattern in the sense of pattern language processing always needs to happen in-line with the regular pattern matching, the actual NESS computations may happen as a post-processing step. When the pattern matching process arrives at a SemSim functional pattern, the symbolic part of the SemSim pattern – i.e. the hyperedge type and the argument roles – are processed immediately, hence in-line. In the case of the contextual variant, the CNESS computations always occur as a post-process. This means that for a given combination of edge and pattern – in which the latter contains at least one occurrence of a `semsim-ctx` sub-pattern – the entire edge and pattern are first matched in regards to everything else. The information necessary for the CNESS measurement is recorded at this point and the actual computations take place afterwards, if everything else matched. This design choice has two advantages: Firstly, the

[3] LevelDB: `https://github.com/google/leveldb`
[4] SQLite: `https://www.sqlite.org/`
[5] spaCy Lexical Tokeniser: `https://spacy.io/usage/linguistic-features#tokenization`

Tokenised Sentence		jerome	eats		an	apple	
Token Indices		0	1		2	3	
Hyperedge	(	eats/P	jerome/C	(	an/M	apple/C	))
Token Position	(	1	0	(	2	3	))

Table 5.1: Example of a token position

context information does not have to be passed down to the SemSim functional pattern matching step and secondly, the computationally expensive CNESS measurements can be avoided, if unnecessary.

SemSim Instances and SemSim Skipping

The recording of the information needed for NESS measurements takes the form of *SemSim Instances*, which can then processed mostly independent from the SH pattern matching. It is possible to configure the pattern matching process via a *SemSim Skipping* option of its software interface, such that the NESS computations are skipped entirely. In this case – also for `semsim-fix` sub-patterns – only the symbolical matching takes place in-line and the SemSim Instances are returned alongside the symbolic matching results. With the skipping option disabled, the `semsim-ctx` related instances are still post-processed as part of the pattern matching process. Isolating the NESS computation by enabling SemSim skipping is especially useful, if the NESS measurements shall be computed with a set of different NESS configuration parameters (e.g. different similarity thresholds), for example in an experimental evaluation.

Token Position Passing

To enable the contextual NESS measurement computations, the *Token Position* has to be provided. In this work, this describes a data structure containing the positions of the tokens corresponding to atoms in a hyperedge. Position here refers to the index of a token in the token sequence, that resulted from the tokenisation of a sentence, when it was translated into a hyperedge (see section 5.1). An example of a tokenised sentence with its corresponding hyperedge and token position is given in table 5.1. The structure of a token position follows the same structure as the hyperedge, which means the token position can be seen as a hyperedge itself (and it is implemented this way). The hypergraph data structure was extended, such that every root edge has an attribute that stores its token position. For the CNESS measurements it is necessary to identify which tokens correspond to the sub-edge that is captured by `semsim-ctx` sub-pattern. Therefore, the token position is passed down alongside the recursive matching process, such that the corresponding sub token position and therefore the relevant token indices can be recorded.

5.2.1 Entry-Point and Pattern Matcher

`graphbrain.patterns.entrypoints`

This module contains all entry-points for pattern matching, primarily serving the hypergraph class. The central entry-point is the `match_pattern` method (see pseudocode 1), which matches a pattern against an edge. This method incorporates a `skip_semsim` argument to control SemSim skipping. A new `Matcher` object (see section 5.2.1) is constructed for every matching procedure (i.e pattern-edge pair). This object maintains the results of the symbolic pattern matching or FNESS extended pattern matching, as well as the SemSim instances that may have been recored due to CNESS extended pattern matching or enabled SemSim skipping. After successful symbolic matching, the `match_semsim_instances` function (see section 5.2.3) is called from here, if SemSim skipping is disabled.

```
def match_pattern(edge, pattern, ref_edges, skip_semsim, hypergraph):
    matcher: Matcher = Matcher(
        edge, pattern, skip_semsim, hypergraph
    )

    if skip_semsim:
        return matcher.results, matcher.semsim_instances

    if matcher.results and match_semsim_instances(
            matcher.semsim_instances,
            pattern,
            edge,
            ref_edges,
            hypergraph
    ):
        return matcher.results

    return []
```

Pseudocode 1: `match_pattern` function

`graphbrain.patterns.matcher`

This module contains the newly created `Matcher` class (see pseudocode 2), designed to function as a stateful object, facilitating the implementation of CNESS extend pattern matching. This object records the symbolic matching results and SemSim instances in its `Matcher.results` and `Matcher.semsim_instances` attributes respectively. It also has an `Matcher.skip_semsim` and a `Matcher.hg` (hypergraph) attribute that store the passed argument values. The classes functionality is primarily realised in the `Matcher.match` method, which serves as the pattern matching recursion entry-point. From this method the `match_semsim` function is called, if a SemSim pattern is encountered.

5.2.2 SemSim Pattern In-Line Processing

The processing of the SemSim functional patterns that occurs in-line with the symbolic matching is implemented in the following submodules of the `patterns.semsim` module:

`graphbrain.patterns.semsim.matching`

This module contains the `match_semsim` function (see pseudocode 3) which is called by a `Matcher` object. In case of FNESS extended pattern matching, the `match_semsim_fix`

```
class Matcher:
    def __init__(self, edge pattern, skip_semsim, hypergraph):
        self.hg = hypergraph
        self.skip_semsim = skip_semsim

        self.semsim_instances = []
        self.results = self.match(
            edge, pattern, current_results=[], tok_pos=get_tok_pos(edge)
        )
```

Pseudocode 2: Pattern `Matcher` class

function (see section 5.2.2) is called – also if SemSim skipping is enabled – to extract the candidate word form the sub-edge and perform the matching with the structural symbolic part of the `semsim-fix` pattern. For CNESS extended pattern matching the pattern is matched symbolically. If the applicable operation of these two returns a *Match*, then a SemSim instance is recorded by the matcher in the case of CNESS extended pattern matching or enabled SemSim skipping.

`graphbrain.patterns.semsim.matching_fix`

This module contains the `match_semsim_fix` function (see pseudocode 4) which is called by the `match_semsim` function. The method first tries to extract the candidate word from the edge, which requires it to be an atom. If the extraction fails, because the edge is not atomic, it is considered not to match. Otherwise, a modified version of the pattern is created, where the pattern content is replaced by the wildcard operator. This modified pattern is then matched against the atomic edge to check if matches structurally (i.e regarding hyperedge type and argument roles). If this is the case and SemSim skipping is disabled, the reference words are extracted from the pattern and the `semsim` function is called, which is the entry-point to the actual NESS measurement implementation.

5.2.3 SemSim Instance Post-Processing

The post-processing of SemSim instances is realised in the following submodules:

`graphbrain.patterns.semsim.instances`

This module primarily defines the `SemSimInstance` data class (see pseudocode 5), which always specifies the SemSim type (i.e. FNESS or CNESS) and the edge that was captured by the corresponding SemSim pattern. Additionally, it contains the candidate word in case of FNESS and the token position in case of CNESS. If the similarity threshold was extracted from the pattern that this SemSim instance was generated from, the instance also stores the threshold.

```
def match_semsim(
    matcher, semsim_function, pattern, edge, current_results, tok_pos
):
    semsim_type, semsim_fix_use_lemma = get_semsim_type(semsim_fun)
    similarity_threshold = extract_similarity_threshold(pattern)

    results = []
    candidate_word = None

    if semsim_type == SemSimType.FIX:
        semsim_fix_results, candidate_word = match_semsim_fix(
            pattern,
            edge,
            current_results,
            matcher.skip_semsim,
            semsim_fix_use_lemma,
            similarity_threshold,
            hypergraph
        )
        if semsim_fix_results is not None:
            results = semsim_fix_results

    if semsim_type == SemSimType.CTX:
        results = matcher.match(edge pattern, current_results, tok_pos)

    if results and (matcher.skip_semsim or semsim_type == SemSimType.CTX):
        add_semsim_instance(
            matcher,
            semsim_type,
            edge,
            candidate_word,
            tok_pos,
            similarity_threshold
        )

    return results
```

Pseudocode 3: `match_semsim` function

`graphbrain.patterns.semsim.references`

This module primarily implements the `get_ref_edges_tok_poses` function (see pseudocode 6), which is required to obtain the token positions of the sub-edges that are captured, when the pattern is matched against the reference edges. SemSim skipping is enabled for this process, since we do not want any NESS measurements to happen, but instead want to access the collected SemSim instances. This function is cached, such that the symbolic matching only occurs once per reference edge and pattern pair.

`graphbrain.patterns.semsim.processing`

This module includes the `match_semsim_instances` function (see pseudocode 7), which acts as the entry-point for post-processing SemSim instances. The function is invoked by

```
def match_semsim_fix(
    pattern, edge, current_results, skip_semsim, use_lemma, threshold, hg,
):
    candidate_word = get_candidate_word(edge, use_lemma, hg)
    if not candidate_word:
        return None, None

    pattern_with_wildcard_content = replace_pattern_content(pattern, '*')
    if not matches_atomic_pattern(edge, pattern_with_wildcard_content):
        return [], None

    if not skip_semsim:
        reference_words = extract_reference_words(pattern)

        if not semsim(
            semsim_type=SemSimType.FIX,
            threshold=threshold,
            cand_word=candidate_word,
            ref_words=reference_words,
        ):
            return [], candidate_word

    return [current_results], candidate_word
```

Pseudocode 4: `match_semsim_fix` function

```
@dataclass
class SemSimInstance:
    type: SemSimType
    edge: Hyperedge
    word: Optional[str] = None
    tok_pos: Optional[Hyperedge] = None
    threshold: Optional[float] = None
```

Pseudocode 5: `SemSimInstance` class

the `match_pattern` entry-point of the pattern matching process (see section 5.2.1), if symbolic matching is successful and SemSim skipping is disabled. Additionally, this function can be accessed by external third-party software that utilise the graphbrain framework to leverage SemSim post-processing.

The arguments of the function are assumed to correspond to each other, such that the produced results are sensible. This means that the given SemSim instances should result from matching the given pattern against the given edge with the SemSim skipping option enabled. Additionally the reference edges should match the pattern structurally. When processing the SemSim instances, the function retrieves the token positions for the reference edges (see section 5.2.3) and collects the necessary token positions to handle a specific *SemSim* instance, which is necessary since the pattern may contain multiple SemSim subpatterns. For each instance, the `semsim` function is called to invoke the NESS measurement computations. If successful for each instance, the edge is considered to match the SemSim instances (and therefore the pattern).

```
@cached
def get_ref_edges_tok_poses(pattern, ref_edges, hypergraph):
    ref_matchers = [
        Matcher(ref_edge, pattern, skip_semsim=True, hypergraph)
        for ref_edge in ref_edges
    ]
    return [instance.tok_pos for instance in matcher.semsim_instances]
```

Pseudocode 6: `get_ref_edges_tok_poses` function

5.3 NESS Measurements: `graphbrain.semsim`

The `graphbrain.semsim` module contains the required functionality for the NESS measurements. The entry-point for this is located in the `graphbrain.semsim.interface` module. The `graphbrain.semsim.matcher` module contains the SemSim matcher classes that perform the actual NESS measurement computations.

5.3.1 SemSim Matcher Interface

`graphbrain.semsim.interface`

This module contains the `semsim` entry-point function, as well as an `init_matcher` function. Additionally it includes the `SemSimConfig` and `SemSimType` class definitions. Core functionality of the module is to maintain different matcher objects that correspond to the different SemSim types, to avoid repeating the computationally expensive initialisation of these objects.

The `semsim` function retrieves the matcher that corresponds to the given SemSim type and passes on the received keyword arguments to this matcher object (see pseudocode 8). A specific matcher object with a specific configuration for a SemSim type can be initialised via the `init_matcher` function, which is publicly exposed. SemSim configurations can be provided via an instance of the `SemSimConfig` data class (see pseudocode 9), which always contains the NESS model name and may also contain a default similarity threshold value. In case of CNESS, the configuration also specifies the usage of the all-tokens option, which is further elaborated in section 5.3.3. The `SemSimType` is an enum class with the members `SemSimType.FIX` and `SemSimType.CTX`, corresponding to FNESS and CNESS.

`graphbrain.semsim.matcher.matcher`

The abstract `SemSimMatcher` class (see pseudocode 10) contained in this module, serves as a unifying interface for the specific matcher implementations that correspond to FNESS and CNESS respectively. Its `SemSimMatcher.similar` method – called by the `semmim` interface function (see pseudocode 8) – calls the `similarities` method of the specific matcher subclass and compares the maximum of the returned similarity values with either the given or the default similarity threshold.

```
def match_semsim_instances(
    semsim_instances, pattern, edge, threshold, ref_words, ref_edges, hypergraph
):
    ref_tok_poses_per_ref_edge = None
    if ref_edges:
        ref_tok_poses_per_ref_edge = get_ref_tok_poses
            pattern, ref_edges, hypgergraph
        )

    for instance_idx, instance in enumerate(semsim_instances):
        threshold = threshold or instance.threshold

        ref_tok_poses = None
        if ref_tok_poses_per_ref_edge:
            ref_tok_poses = [
                ref_tok_poses[instance_idx]
                for ref_tok_poses in ref_tok_poses_per_ref_edge
            ]

        if not semsim(
            semsim_type=instance.type,
            threshold=threshold,
            cand_word=instance.word,
            ref_words=ref_words,
            cand_edge=edge,
            cand_tok_pos=instance.tok_pos,
            ref_edges=ref_edges,
            ref_tok_poses=ref_tok_poses,
            hypergraph=hypergraph
        ):
            return False

    return True
```

Pseudocode 7: `match_semsim_instances` function

5.3.2 Fixed Word Embedding Matcher

The `graphbrain.semsim.matcher.fixed` module implements the fixed word neural embedding based semantic similarity measurement in form of the `FixedEmbeddingMatcher` class (see pseudocode 11), which inherits from the `SemSimMatcher` class.

Fixed word embedding models typically operate with a predefined vocabulary, necessitating an initial check to ensure that the candidate word and the reference words are included. As mentioned in section 2.3.1, certain models can handle unknown words by employing subword tokens, allowing them to generalise beyond their initial vocabulary. Should embeddings be available for all relevant words, the model can then calculate the pairwise similarity between the candidate word and the reference words.

The established NLP library *gensim* (Řehůřek and Sojka 2010) is employed to utilise fixed word embedding models, providing a unifying interface for various different models. Cosine similarity between two words can be computed using the `model.similarity` method, assuming the `model` has been initialised through *gensim*. Apart of subword-based models,

```
def semsim(
    semsim_type, **keyword_arguments
):
    matcher = get_matcher(matcher_type=semsim_type)
    return matcher.similar(**keyword_arguments)
```

Pseudocode 8: `semsim` function

```
@dataclass
class SemSimConfig:
    model_name: str
    similarity_threshold: Optional[float] = None
    use_all_tokens: Optional[bool] = None
```

Pseudocode 9: `SemSimConfig` data class

a trained fixed neural word embedding model does not require complex computational operations to generate embeddings. Instead, it is fundamentally realised as a key-value storage of words and their corresponding embeddings.

Fixed Word Neural Embedding Models

Two specific fixed word neural embedding models have been selected from the models, which are available through gensim[6] to be utilised in this work. When referring to the simplified model names in the following chapters, we are referring to the specific gensim models corresponding to the identifiers stated below.

word2vec The `word2vec-google-news-300` model is chosen as the baseline embedding model for fixed word NESS and therefore for NESS in general. This specific variant of the word2vec model was trained using the skip-gram negative-sampling strategy (see section 2.3.1) on a Google News corpus, encompassing approximately 100 billion words.

conceptnet-numberbatch The `conceptnet-numberbatch-17-06-300` model is selected to represent the state-of-the-art of fixed word neural embedding models. Although the MetaVec model has been identified to outperform Conceptnet Numberbatch in our literature review (see section 2.3.1), this model is not available through gensim. Furthermore, it shall be noted that the selected model corresponds to version 17.06 of Conceptnet Numberbatch, while the most recent – and reportedly better performing – iteration of the model is version 19.08.[7] We utilize the older version of the model, because the newer version is not available through gensim.

The aforementioned limitations in model performance are accepted in favour of implementation simplification. They are considered tolerable for this work, since our primary

[6] gensim data repository: `https://github.com/piskvorky/gensim-data`

[7] Conceptnet Numberbatch 19.08:
`http://blog.conceptnet.io/posts/2019/conceptnet-numberbatch-19-08/`
(Accessed on Oct. 9th, 2023)

```
@abstractclass
class SemSimMatcher:
    def __init__(self, config):
        self.similarity_threshold = config.similarity_threshold

    def similar(self, threshold, **keyword_arguments):
        similarities = self.similarities(**keyword_arguments)
        if not similarities:
            return False

        similarity_threshold = threshold or self.similarity_threshold
        if max(similarities) < similarity_threshold:
            return False

        return True

    @abstractmethod
    def similarities(self, **keyword_arguments):
        raise NotImplementedError
```

Pseudocode 10: Abstract SemSimMatcher class

```
class FixedEmbeddingMatcher(SemSimMatcher):
    def __init__(self, config):
        super().__init__(config=config)
        self.model = load_model(config.model_name)

    def similarities(self, cand_word, ref_words):
        if not self.in_vocab([cand_word] + ref_words):
            return None

        return [
            self.model.similarity(cand_word, ref_word) for ref_word ref_words
        ]
```

Pseudocode 11: FixedEmbeddingMatcher class

interest in the model selection is not to produce state-of-the-art results, but to compare the performance of different embedding models in the context of NESS-SHPM.

5.3.3 Contextual Embedding Matcher

The `graphbrain.semsim.matcher.context` module realises the contextual neural embedding based semantic similarity measurement in form of the `ContextEmbeddingMatcher` class (see pseudocode 12), which inherits from the `SemSimMatcher` class. Computing the contextual neural embedding-based semantic similarity involves multiple steps as outlined in section 4.2.3, which are in the following presented in there specific implementation.

First, the tokens that correspond to the candidate and reference edges are retrieved – in the following referred to as *candidate tokens* and *reference tokens*. It shall be noted here, that the candidate edge is not the sub-edge that was captured by the SemSim pattern triggering this CNESS measurement. Instead, it is the original matching target of the pattern, which was given to the top-level `match_pattern` entry-point (see section 5.2.1). The candidate edge as well as the reference edges need to be root edges, i.e. they represent entire sentences in the text corpus based on which the hypergraph was created. In the hypergraph data structure, every root edge has an attribute containing its corresponding token sequence (see section 5.1).

Secondly, the token indices are extracted from the *candidate token position* and *reference token positions*, which correspond to the sub-edges that were captured by the SemSim pattern. The candidate token position was passed down to the `match_semsim` function (see pseudocode 3) and was stored in the SemSim instance, while the reference token position was retrieved via matching of the reference edges against the pattern (see pseudocode 6). The extracted token indices are in the following referred to as *lexical indices*, since they correspond to the lexical tokenisation via spaCy. If the *all-tokens* option is set (see section 5.3.1), the lexical token indices are not extracted from the token positions, but encompass all indices of the lexical token sequence.

Given the candidate and reference tokens, as well as the relevant lexical indices, we have all information needed to compute the contextual embeddings, specific to the lexical tokens that are specified by the lexical token indices. To elaborate on the computing procedure, we introduce the contextual neural embedding models that are utilised in this work.

Contextual Neural Embedding Models

Two specific contextual neural embedding models have been selected to be utilised in this work. Both models are Transformer-based and can be instantiated via the established NLP library *transformers* (Wolf et al. 2020). When referring to the simplified model names in the following chapters, we mean the specific models presented below.

GTE This model has been chosen, because it was identified to show stat-of-the-art performance among contextual neural embedding models in our literature review (see section 2.3.2). It is used in its *GTE-large*[8] variant, which is the largest and best performing variant available.

E5 This model has been selected because it was found to perform competitively compared to GTE (see section 2.3.2) and shares important implementation intricacies. It is utilised in its *E5-large-v2*[9] variant, which is the largest variant of the second version of the model family and overall the best performing variant of the model.

Since both models are based on BERT, they also both make use of its tokenizer, the *WordPiece* tokenizer (Devlin et al. 2019). Another commonality is the application of the same average pooling procedure to construct the final embedding from the *token embeddings* (see pseudocode 13). In the following, these token embeddings do not correspond to lexical tokens, but to tokens produced by the WordPiece tokeniser.

[8] GTE-large: `https://huggingface.co/thenlper/gte-large`

[9] E5-large-v2: `https://huggingface.co/intfloat/e5-large-v2`

```
class ContextEmbeddingMatcher(SemSimMatcher):
    def __init__(self, config: SemSimConfig):
        super().__init__(config=config)
        self.use_all_tokens: bool = config.use_all_tokens
        self.spacy_pipe: Language = create_spacy_pipeline(config.model_name)

    def similarities(
            self, cand_edge,cand_tok_pos, ref_edges, ref_tok_poses, hypergraph
    ):
        cand_tokens = get_edge_tokens(cand_edge, hg)
        refs_tokens = [get_edge_tokens(ref_edge, hg) for ref_edge in ref_edges]

        cand_tok_idxes = get_tok_idxes(
            cand_tok_pos, len(cand_tokens), self.use_all_tokens
        )
        refs_tok_idxes = [
            get_tok_idxes(ref_tok_pos, len(ref_tokens), self.use_all_tokens)
            for ref_tok_pos, ref_tokens in zip(ref_tok_poses, refs_tokens)
        ]

        cand_embedding = self.get_embedding(cand_tokens, cand_tok_idxes)
        refs_embeddings = [
            self.get_embedding(ref_tokens, ref_tok_idxes)
            for ref_tokens, ref_tok_idxes in zip(refs_tokens, refs_tok_idxes)
        ]

        return [
            cosine_similarity(cand_embedding, ref_embedding)
            for ref_embedding in refs_embeddings
        ]

    @cached
    def get_embedding(self, tokens, tok_idxes):
        trf_data = self.spacy_pipe(tokens).trf_data
        return get_trf_embedding_of_lex_tokens(trf_data, tok_idxes)
```

Pseudocode 12: `ContextEmbeddingMatcher` class

```
def get_trf_embedding_of_lex_tokens(trf_data, lex_tok_idxesn):
    trf_tok_idxes = get_trf_tok_idxes(lex_tok_idxes, trf_data.alignment)
    trf_embeddings = trf_data.model_output.last_hidden_state[trf_tok_idxes])
    return average_pool(trf_embeddings, trf_data.attention_mask[trf_tok_idxes]))

def average_pool(last_hidden_states, attention_mask):
    last_hidden = last_hidden_states.masked_fill(attention_mask.bool(), 0.0)
    return last_hidden.sum() / attention_mask.sum()
```

Pseudocode 13: `get_trf_embedding_of_lex_tokens` and `average_pool` functions

Token Alignment

Central to the lexical token specific embedding computation is the creation of a modified *spaCy transformer pipeline*. This means inserting a custom model in place of the default model of the Transformer-based spaCy pipeline, which itself is built upon the transformers library. We do not utilise the models via transformers directly, because the application through the spaCy pipeline provides the necessary token alignment information.

We have to align the lexical token indices with the corresponding token indices of the WordPiece tokenisation. After passing the tokens through the spaCy transformer pipeline, we can retrieve the *transformer data*, which contains the token embeddings and the token *alignment data* (see pseudocode 13) – itself consisting of *alignment indices* and *alignment lengths*. The information that these data objects contain is illustrated by the example in table 5.2.

S:	Don't go into the devil's H.Q.														**Length**
LT:	Do		n't		go	into	the	devil	's			H.Q.			8
WT	don		'	t	go	into	the	devil	'	s	h	.	q	.	13
AI:	0	0	1	2	3	4	5	6	7	8	9	10	11	12	14
AL:	1	3			1	1	1	1	2		4				8

Table 5.2: Example of token alignment: Sentence (**S**), Lexical Tokenisation (**LT**), WordPiece Tokenisation (**WT**), Alignment Indices (**AI**), Alignment Length (**AL**)

Alignment lengths (AL) is a list that specifies for every lexical token index, how many *transformer indices* correspond to it. Transformer indices refer to indices of tokens in the WordPiece tokenisation. Alignment indices (AI) is a list that contains transformer indices. In the following an element at index i of a list L will be referred to as L_i. To identify which transformer indices I_{trf} correspond to a lexical token index i_{lex}, we have to take the alignment indices from index i_{sum} to index $i_{\text{sum}}+AL_{i_{\text{lex}}}$, where $i_{\text{sum}} = \sum_{j=0}^{i_{\text{lex}}-1} AL_j$ is the sum of all alignment lengths before $AL_{i_{\text{lex}}}$. This results in following formula:

$$I_{\text{trf}}(i_{\text{lex}}) = \{AI_k \mid k \in \{ \sum_{j=0}^{i_{\text{lex}}-1} AL_j, ..., \sum_{j=0}^{i_{\text{lex}}-1} AL_j + AL_{i_{\text{lex}}} - 1\}\}$$

Based on this formula, the token alignment procedure is implemented (see pseudocode 14). It should be noted, that a transformer index may occur multiple times in the alignment indices. That happens if a transformer tokens corresponds to multiple lexical tokens, as can be seen for the first token in the example above (see table 5.2). Since we are not interested in the reverse mapping, we circumvent this problem by checking if a transformer index is already contained in I_{trf} before appending it, while building the list iteratively.

Embedding Computation

By invoking the `ContextMatcher.get_embedding` method with the entire lexical token sequence and the relevant lexical indices, the computation of the embeddings is initiated. This method is cached, so that an embedding has to be computed only once for a pair of token sequence (i.e. root edge) and indices subset. It shall be noted, that the input of the

```
def get_trf_tok_idxes(lex_tok_idxes, alignment):
    lex2trf_idx = get_lex2trf_tok_idx(alignment)
    trf_tok_idxes = []
    for lex_tok_idx in lex_tok_idxes:
        for trf_idx in lex2trf_idx[lex_tok_idx]:
            if trf_idx not in trf_tok_idxes:
                trf_tok_idxes.append(trf_idx)
    return trf_tok_idxes


def get_lex2trf_tok_idx(alignment):
    lex2trf_idx = {}
    trf_idx = 0
    for trf_tok_len in alignment.lengths:
        lex2trf_idx[lex_idx] = alignment.indices[trf_idx:trf_idx + trf_tok_len]
        trf_idx += trf_tok_len
    return lex2trf_idx
```

Pseudocode 14: `get_trf_tok_idxes` and `get_lex2trf_tok_idx` functions

spaCy transformer pipeline – and therefore the contextual neural embedding model – is always the entire token sequence, since it is the context of the relevant tokens.

The model produces token embeddings of which the relevant subset is selected. This subset is identified by passing the relevant lexical token indices to the token alignment procedure described above, which results in corresponding transformer indices. Technically, the token embeddings are the states of the last hidden layers of the contextual neural embedding model. Average pooling the hidden states corresponding to the relevant transformer indices, yields the final contextual embedding. This embedding is specific to the lexical token indices that were given and consequently it is specific to the sub-edge that was captured by the `semsim-ctx` pattern, if the all-tokens option is not set (see pseudocode 13).

The embeddings are computed for the candidate edge and reference edges and their pairwise cosine similarities are measured (see pseudocode 12).

6 Evaluation

In this chapter the conceived concept (see chapter 4) and specific implementation (see chapter 5) of the neural embedding-based semantic similarity extended Semantic Hypergraph pattern matching system is evaluated to answer research questions posed in section 3.1. Therefore a a case study is conducted to evaluate the system for a specific use case.

The software code specific to the evaluation is available in a separate repository.[1]

6.1 Case Study: Conflicts

The conflicts case study follows the approach presented in Menezes and Roth 2021, where expressions of conflict are extracted from a given SH using a single SH pattern. In their work they build upon the information extracted by the pattern to conduct further analyses, which are not in the scope of this work. Here the evaluation is limited to the task of classifying whether the content of a given edge in the SH is an expression of conflict or not. Or framed differently, the task is to retrieve exactly all those edges whose content is an expression of conflict. The evaluation will compare the retrieval performance of a suitable set of different SH patterns and corresponding configuration of the NESS-SHPM system by matching them against a labelled dataset of hyperedges.

6.1.1 Expressions of Conflict

An expression of conflict in the context of this case study is defined as a sentence which fulfils the following properties:

> There is a conflict between two explicitly named actors, wherever these actors are mentioned in the sentence; whereby a conflict is defined as antagonizing desired outcomes.

6.1.2 Reddit Worldnews Corpus

The corpus from which those expressions of conflict are retrieved consists of news titles that were shared on the social media platform *Reddit*. Specifically all titles shared between January 1st, 2013 and August 1st, 2017 on *r/worldnews*, which is described as: “A place for major news from around the world, excluding US-internal news.”[2] This corpus contains 479,384 news headers and is in the following referred to as the *Worldnews-Corpus*.

Each of these headers is comprised of a single sentence and is forms a root edge in the SH constructed from it, which is referred to as the *Worldnews-SH*. Parsing errors, which may

[1] Thesis evaluation implementation: `https://github.com/wuxmax/graphbrain-semsim-thesis`

[2] `http://reddit.com/r/worldnews`

potentially have occurred during this translation are out of scope of this work. All edges in the Worldnews-SH are assumed to be correctly parsed.

6.1.3 Semantic Hypergraph Patterns

The SH patterns that are used in this evaluation all have the same general form, to isolate the effect of replacing a purely symbolic matching against a specific word or list of words with NESS-SHPM. In this section the general form of these patterns will be described, which entails consequences for the creation of the labelled dataset described in section 6.2.

Original Conflict Pattern

Pattern 7 is originally defined in Menezes and Roth 2021, p. 22. We use a rewritten, but functionally identical form of it, which is shown in pattern 8 and is in the following referred to as the *original conflict pattern.* It is used to extract conflicts between two parties SOURCE and TARGET, potentially regarding some TOPIC. As mentioned before, the assignment of these variables is irrelevant for this case study.

The original conflict patterns contains two sub-patterns which utilize word lists. These sub-patterns match the trigger sub-edge and predicate sub-edge of a candidate edge respectively and are in following referred to as *trigger sub-pattern* and *predicate sub-pattern.* If not stated otherwise these terms will refer to pattern 8.

- **Trigger sub-pattern:** [against,for,of,over]/T
- **Predicate sub-pattern:** (lemma >PRED/P.{so,x}
  [accuse,arrest,clash,condemn,kill,slam,warn])

In the trigger sub-pattern the content of the candidate trigger sub-edge is directly matched against a list of prepositions, which are in the following referred to as the *conflict triggers.* In case of the predicate sub-pattern, the word list is matched against the lemma of the innermost atom of the candidate predicate sub-edge, which is always a verb. The list of verbs used here will in the following be referred to as the *conflict predicates.*

- **Conflict triggers:** against, for, of, over
- **Conflict predicates:** accuse, arrest, clash, condemn, kill, slam, warn

(PRED/P.{so,x} SOURCE/C TARGET/C [against,for,of,over]/T TOPIC/[RS]) ∧
(lemma/J >PRED/P [accuse,arrest,clash,condemn,kill,slam,warn]/P)

Pattern 7: Original conflict pattern

((lemma PRED >[accuse,arrest,clash,condemn,kill,slam,warn]/P.{so,x})
SOURCE/C TARGET/C [against,for,of,over]/T TOPIC/[RS])

Pattern 8: Original conflict pattern (rewritten)

Wildcard Conflict Patterns

Replacing either the trigger sub-pattern, the predicate sub-pattern or both of them with a semsim function are the options for utilizing NESS-SHPM in a modified version of pattern 8 without modifying the general structure of the pattern. To evaluate which of these options are best suited to evaluate the retrieval performance of NESS-SHPM, two *wildcard conflict patterns* are constructed. In these patterns the predicate sub-pattern (pattern 9) or the trigger sub-pattern (pattern 10) are replaced by the wildcard operator.

(*/P.{so,x} SOURCE/C TARGET/C [against,for,of,over]/T TOPIC/[RS])

Pattern 9: Predicate wildcard pattern

((lemma PRED/P.so,x [accuse,arrest,clash,condemn,kill,slam,warn]) SOURCE/C TARGET/C */T TOPIC/[RS])

Pattern 10: Trigger wildcard pattern

Preliminary Evaluation The original conflict pattern and the two wildcard conflict patterns are matched against the Worldnews-SH and the number of matches is recorded. Comparing the number of matches of these patterns shows which of the sub-patterns is most influential for the retrieval performance of pattern 8. Table 6.1 shows the results of these preliminary evaluations. It can be seen that the choice of conflict predicates is much more influential on the number of matches than the choice of conflict triggers when compared to the number of matches resulting from the original conflict pattern. While replacing the predicate sub-pattern with a wildcard operator yields an increase with a factor of $12,45$, replacing the trigger sub-pattern with a wildcard operator only yields an increase with a factor of $1,07$.

Pattern name	Number of matches
Original conflict pattern	5766
Predicate wildcard pattern	71804
Trigger wildcard pattern	6154

Table 6.1: Results of matching the original conflict pattern and the wildcard patterns against the Worldnews-SH

SemSim Conflict Patterns

Based on the result of the preliminary evaluation in section 6.1.3, the predicate sub-pattern of pattern 8 is replaced by different forms of semsim functional patterns to construct different *semsim conflict patterns*. These patterns are then used to evaluate the effects of utilizing NESS-SHPM. The trigger sub-pattern is not modified to better isolate these effects in comparison to purely symbolic SHPM.

Pattern 11 describes the general form of a semsim conflict pattern. The `<SEMSIM-FUNCTION>` placeholder is replaced with one of the three implemented semsim functions to construct the *semsim-fix conflict pattern* (pattern 12), *semsim-fix-lemma conflict pattern* (pattern 13) and the *semsim-ctx conflict pattern* (pattern 14). The conflict predicate list is used as as `<REFERENCE-CONTENT>`, in pattern 12 and pattern 13, which both utilize FNESS. In the semsim-ctx conflict pattern, no `<REFERENCE-CONTENT>` is specified, since the variable itself matches any content is therefore functionally identical to a wildcard operator in this context. The necessary reference edges can only be provided via an external argument and not inside the pattern.

```
( ( <SEMSIM-FUNCTION> PRED/P.so,x <REFERENCE-CONTENT> )
SOURCE/C TARGET/C [against,for,of,over]/T TOPIC/[RS] )
```

Pattern 11: General SemSim conflict pattern

```
( ( semsim-fix >PRED/P.so,x [accuse,arrest,clash,condemn,kill,slam,warn] )
SOURCE/C TARGET/C [against,for,of,over]/T TOPIC/[RS] )
```

Pattern 12: semsim-fix conflict pattern

```
( ( semsim-fix-lemma >PRED/P.so,x [accuse,arrest,clash,condemn,kill,slam,warn] )
SOURCE/C TARGET/C [against,for,of,over]/T TOPIC/[RS] )
```

Pattern 13: semsim-fix-lemma conflict pattern

```
( ( semsim-ctx PRED/P.so,x ) SOURCE/C TARGET/C [against,for,of,over]/T TOPIC/[RS] )
```

Pattern 14: semsim-ctx conflict pattern

6.2 Conflict Dataset

To conduct an evaluation which assesses the retrieval performance of the NESS-SHPM system, it is necessary to have a dataset of edges with labels that state whether an edge is an expression of conflict or not. Since such a dataset does not exist it needs to be constructed. In the following the construction process of this *Conflict Dataset* (CD), which is used for the evaluation in this case study, and the datasets characteristics are discussed.

6.2.1 Base Edge Set

The set of edges that can be retrieved by a conflict pattern, i.e. the original conflict pattern or a semsim conflict pattern, is restricted the general form of these patterns. This entails that, given the same SH, every set of matching edges of a pattern of this form will be a subset of the matching edges of the predicate wildcard pattern (pattern 9). The set of edges resulting form matching this pattern against the Worldnews-SH are therefore used as the *Base Edge Set* (BES) from which the conflict dataset is constructed, instead of the entirety of all the hypergraphs root edges.

Predicate Lemma Every edge in the BES has a predicate sub-edge that has an innermost atom, which is a verb that has a lemma. In the following this is called the *predicate lemma* of an edge. Each of the edges matching pattern 8 or a pattern in the form of pattern 11 therefore corresponds to a predicate lemma.

6.2.2 Desired Characteristics

To effectively evaluate the effectiveness of the application of NESS by matching a pattern in the form of pattern 11, the dataset used for this should have the following characteristics:

- Contain the largest possible number of unique predicate lemmas
- Contain the largest possible number of edges per unique predicate lemma

On the one hand it is desired to have as many different unique predicate lemmas as possible in the dataset to be able to evaluate whether NESS can differentiate if a predicate lemma indicates an expression of conflict or not. On the other hand it is desired to have as many different edges per unique lemma as possible in the dataset to be able to evaluate whether CNESS is able to differentiate if edges represent an expression of conflict or not, given that they correspond to the same predicate lemma.

6.2.3 Construction Process

To create the labelled CD, the edges of the dataset need to be manually labelled by human annotators, which is labor-intensive. The BES contains $n_b = 71804$ edges. Due to the time constrains of this work and the limited availability of three annotators, the BES needs to be subsampled to create the CD.

Filtering

Since the desired characteristics described above relate the the distribution of predicate lemmas, it is relevant to verify that is possible to determine the predicate lemma for all edges in the edge set from which the CD is sampled. In some cases it is not possible to determine the predicate lemma of a given edge due to implementation issues, which out of scope of this work. In these cases an edge is filtered from the BES, which results in the *Filtered Base Edge Set* (FBES). The FBES contains $n_f = 69380$ edges.

Sampling

The edges in the FBES correspond to $n_l = 2195$ unique predicate lemmas. Attaining to the desired dataset characteristics, the number of samples n_s in the subsampled dataset should ideally be a multiple $m_l >= 2$ of n_l, so that $n_s = m_l \cdot n_l$. This would mean that every predicate lemma contained in the FBES is statistically represented multiple times in the subsampled dataset. A dataset size of $n_s = 2000$ was chosen, which means $m_l < 2$ and $n_s << n_f$. This entails that a trade-off between the desired dataset characteristics has to be made. To account for this, a sampling method is applied that offers more control over the distribution of predicate lemmas in the subsampled dataset than uniform random

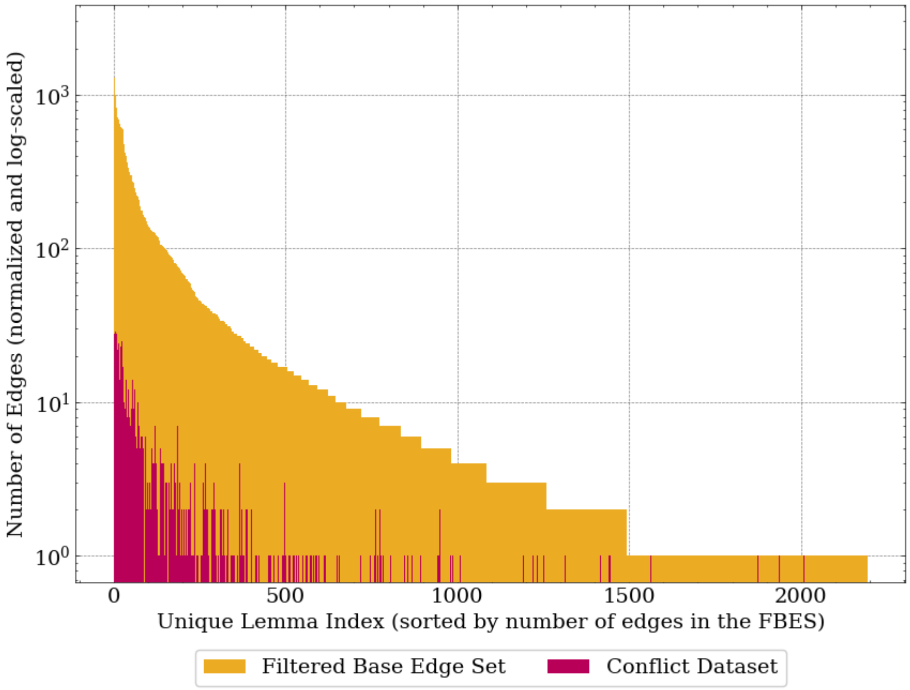


Figure 6.1: Distribution of unique lemmas in the FBES and CD

sampling does. This sampling method is based on the idea of *Stratified Sampling* (Parsons 2017) and is described in detail in algorithm 1.

The procedure splits the FBES into multiple bins after the edges are sorted by number of occurrence of their predicate lemma and then uniformly randomly samples from each bin. This method guarantees that predicate lemmas which correspond to a relatively small number of edges in the FBES will be represented in the subsampled dataset. The distribution of uniques lemmas in the FBES and the CD is compared visually in fig. 6.1.

Algorithm 1 Dataset sampling algorithm

```
1. Create a list of tuples $t$ of edges and their corresponding predicate lemma:
   $L = [(l_k, e_i), ...]$ with $k \in \{0, ..., m\}$ and $i \in \{0, ..., n\}$
2. Sort this list by the number of tuples containing a predicate lemma to create the list:
   $L_{sort} = [(l_0, e_0), ...(l_m, e_n)]$, so that:
     • $n_k$ is the number of tuples containing a lemma $l_k$
     • $t_j$ with $j > i$ is a tuple wich sorted after tuple $t_i$
     • $n_o >= n_p$ if $t_i = (l_o, e_i)$ and $t_j = (l_p, e_j)$
3. Split the list $L_{sort}$ into $n_b$ bins.
4. Uniformly sample $n_{sb}$ tuples from each bin.
5. Build a set of all edges $e$ contained in the sampled tuples.
```

The subsampled dataset size resulting from this sampling method is $n_s = n_b * n_{sb}$. Given $n_s = 2000$, the values $n_b = 10$ and $n_{sb} = 200$ were chosen for sampling the CD.

Edge set name	Number of all edges	Number of un. lemmas	Number of conflict edges (% of all edges)	Number of no conflict edges (% of all edges)
Worldnews-SH	479384	-	-	-
Base Edge Set (BES)	71804	-	-	-
Filtered BES (FBES)	69380	2195	-	-
Conflict Dataset (CD)	2000	539	599 (29.95 %)	1401 (70.05 %)

Table 6.2: Number of edges, number of unique lemmas and proportion of labels for the different edge sets

Labelling

The labelling task is shared between the three annotators. A given edge will be either labeled as *conflict* or *no conflict* by an annotator following the definition given in section 6.1.1. Because of the aforementioned time constraints, every edge is only labeled by one annotator. To nonetheless ensure a consistent labelling among all annotators, a set of 50 edge is labelled by all three annotators. Every edge for which a disagreement in labelling occurs between at least two of the annotators, is inspected to reach an agreement on the label. Utilizing this process, the annotators understanding of what constitutes an expression of conflict is refined. Following this preliminary step, the n_s edges of the dataset are equally distributed among the three annotators and individually labelled by them.

6.2.4 Edge Set Comparison

The CD is the result of the filtering, sampling and labelling described above. The size of the Worldnews-SH, BES, FBES and CD are listed in table 6.2 for comparison. If applicable, the number of unique lemmas as well as the number and percentage of edges which are labelled as an expression of conflict and of those which are not are also noted. Table A.1 in appendix A.1 list the content of randomly sampled exemplary edges for the Worldnews-SH, the FBES and the CD. For the latter, the corresponding edge labels are also listed.

6.3 Evaluation Process

In this evaluation multiple *evaluation runs* are conducted. Each evaluation run correspond to a SHPM process in which a pattern is matched against the CD. In the case of patterns utilizing NESS this requires that additional parameters in form of an *NESS configuration* are given to the matching process. An evaluation run is described by an *evaluation (run) configuration*. For each evaluation run the *evaluation metrics* are computed.

6.3.1 Evaluation Run Configurations

An evaluation configuration consists of the following parameters:

- Conflict Pattern
- NESS Configuration (in case of NESS-SHPM):

- NESS model
- Similarity Threshold
- Use all tokens (in the case of CNESS)
- Reference Edge Set (in the case of CNESS)

Conflict Patterns The four conflict patterns used in this evaluation are described in detail in section 6.1.3. An overview of the properties of these patterns can be seen in table 6.3.

NESS models The neural embedding model – or NESS model – may be one of those described in section 5.3.2 and section 5.3.3: *word2vec*, *conceptnetnumber-batch*, *E5* and *GTE*. In case of a FNESS utilizing conflict pattern, the NESS model name is added in its shortened form: word2vec as *w2v* and conceptnet-numberbatch as *cn*

Similarity Thresholds In this evaluation the similarity threshold t_s is always selected from a range of thresholds $r_t = \{0, 0.01, ..., 0.99, 1.00\}$, i.e. $t_s \in r_t$. This results in 101 different values of t_s.

Reference Edge Sets Multiple *Reference Edge Set* (RES) are randomly sampled from the set of edges in the CD, which are labelled as "conflict". These edges are then excluded from the dataset, to avoid introducing data from the test dataset to the system that is being evaluated. To compare the effect of different sample sizes, differently sized sets are drawn. To compare the effect of different samples, different samples are drawn. A RES with ID *N-X* has $N \in \{1, 3, 10\}$ samples and is from sample draw $X \in \{1, 2, 3, 4, 5\}$. This results in 15 different RESs in total. The specific sets that have been sampled can be seen in appendix A.2.

Evaluation Run Names

An evaluation run name has the form: `CP NM-AT r-N-X t-TS`

In a specific evaluation run name, the placeholders (capitalised letters) are replaced with actual values. Such a name always begins with the conflict pattern (CP) name in its shortened form: *original*, *semsim-fix*, *semsim-fix-lemma* or *semsim-ctx*. This is followed by the NESS model (NM) name. If the NESS type is CNESS, the usage of the all-tokens (AT) option is indicated by adding `-at` to the model name. Also in case of CNESS, the reference edge set ID N-X is indicated by appending `r-N-X`. For all NESS utilizing evaluation runs, the similarity threshold t_s is indicated by appending `t-TS`, where `TS` is the value of t_s.

Pattern name	Lemma based	NESS type	Includes ref. words	Requires ref. edges
Original conflict pattern (8)	Yes	-	-	-
semsim-fix conflict pattern (12)	No	Fixed	Yes	No
semsim-fix-lemma conflict (13)	Yes	Fixed	Yes	No
semsim-ctx conflict patter (14)	No	Contextual	No	Yes

Table 6.3: Properties of the conflict patterns used in the evaluation

Specification of Evaluation Run Configurations

The configurations for all evaluation runs that are conducted in this case study are specified in table 6.4. All possible parameter combinations of an evaluation configuration, i.e. the conflict pattern and the NESS parameters, are evaluated. The total number of conducted evaluation runs therefore amounts to 6465.[3] In table 6.4 the different values for the reference edge set ID and the ST are omitted. The *random* evaluation run configuration relates to a hypothetical evaluation run in which edges are uniformly randomly matched.

6.3.2 Evaluation Metrics

Using the information provided by the dataset labels it is determined whether a match is correct or not. If an edge is matches in a given evaluation run and is labeled as "conflict" in the dataset, it is considered a *true positive* (TP). If an edge matches but is labeled "no conflict", it is considered a *false positive* (FP). The *true negatives* (TN) and *false negatives* (FN) are determined analogously by examining the non-matching edges. Based on the TP, FP, TN and FN the metrics *precision*, *recall* and *F1-score* are computed.

Jointly inspecting precision and recall of a system allows to asses its *retrieval performance*, since there generally exists a trade-off between these system properties. The F1-Score is an established metric, which integrates precision and recall and therefore acts as an measurement of overall retrieval performance in this evaluation.

Relationship of Similarity Threshold and Recall It can be generally stated that the recall (r) of NESS-SHPM in relation to the similarity threshold ($r(t_s)$) is strictly monotonically decreasing, since the set of points in embedding space that is inside of the similarity boundary consistently gets smaller with increasing threshold.

6.4 Evaluation Results

In this section the results of the evaluation runs which are defined by the evaluation configurations in section 6.3.1 are examined. Different perspectives on the result data are constructed in the form of tables and plots to enable answering the research questions. The following subsections each represent one perspective and conclude with significant observations that can be made based on it.

[3] 1 (original) + 2 (semsim-fix and semxim-fix-lemma) * 2 (FNESS models) * 101 (STs)
+ 1 (semsim-ctx) * 2 (CNESS models) * 2 (all tokens) * 15 (ref. edge sets) * 101 (STs)

Evaluation Run Name	Conflict Pattern	NESS Configuration	
		NESS Model	all tokens
original	original	-	-
semsim-fix w2v	semsim-fix	word2vec	-
semsim-fix cn	semsim-fix	conceptnet-numbatch	-
semsim-fix-lemma w2v	semsim-fix-lemma	word2vec	-
semsim-fix-lemma cn	semsim-fix-lemma	conceptnet-numbatch	-
semsim-ctx e5 r-N-X	semsim-ctx	E5	No
semsim-ctx gte r-N-X	semsim-ctx	GTE	No
semsim-ctx e5-at r-N-X	semsim-ctx	E5	Yes
semsim-ctx gte-at r-N-X	semsim-ctx	GTE	Yes

Table 6.4: Evaluation Run Configurations

Result Data Description Concepts

To facilitate constructing insightful perspectives on the result data, some novel concepts for its description are introduced in the following.

Evaluation Run Sets Multiple evaluation runs can be grouped into an *Evaluation Run Set* (ERS) according to their shared configuration parameter values. The naming convention for an ERS follows the evaluation run naming convention described in section 6.3.1. The parameters values that are not shared among the evaluation runs in the ERS are omitted from the name or replaced by the wildcard symbol `*`. The placeholders (capitalised letters) are used to refer to an ESR of a generic form with fixed parameter values without specifying these values. By surrounding a part of the ERS name with parentheses `(*)`, it is indicated that this part is omitted if unsuitable.

Examples are given to illustrate this:

- An ERS of all evaluation runs utilizing any form of NESS with $t_s = 0.5$ is named: `semsim-* t-0.5`
- An ERS of all evaluation runs utilizing FNESS with an unspecified but fixed NESS model has the form: `semsim-fix(-*) NM`
- An ERS of all evaluation runs utilizing NESS with all parameters (that are applicable) fixed but unspecified, except for the specific value $t_s = 0.5$, has the form: `semsim-* NM(-AT) (r-N-X) t-0.5`

Best F1-Score Evaluation run The *best F1-Score evaluation run* refers to the evaluation run with the highest F1-Score in an ERS corresponding to a NESS utilizing evaluation configuration where every parameter except for t_s is fixed. Such an ERS is generally named `semsim-* NM(-AT) (r-N-X)`. The corresponding F1-score is also simply referred to as *best F1-score*.

Mean Reference Edge Set Evaluation Runs A *mean reference edge set evaluation run* is constructed from the mean value of all evaluation scores for the evaluation runs in an ERS of the form `semsim-ctx NM-AT r-N-* t-TS`. This means for every t_s the mean of the corresponding evaluation scores of all reference edge sets of the same size is computed. In the following these synthetical evaluation runs are referred to in this form: `semsim-ctx NM-AT r-N-mean`

Mean Reference Edge Set Best F1-Score Evaluation Metric Scores The *mean reference edge set (RES) best F1-Score evaluation metric scores* are the mean values of all evaluation scores corresponding to the best F1 score for all evaluation runs in an ERS of the form `semsim-ctx NM-AT r-N-*`. In the following these evaluation metric scores will be referred to in this form: `semsim-ctx NM-AT r-N-mean-best`

6.4.1 Best F1-Score based Evaluation Run Comparison

Table 6.5 shows the evaluation scores for all evaluation metrics of the best F1-score evaluation runs for the original conflict pattern evaluation run and all evaluation runs utilizing FNESS. For the evaluation runs utilizing CNESS only the mean RES best F1-score evaluation metric scores and the standard deviation of the best F1-scores for the corresponding ERSs are shown. The t_s value listed for the mean RES best F1 score evaluation metrics is the mean of all t_s values for the best F1-scores in the corresponding ERSs.

In table A.6 and table A.7 of appendix A.3 the best F1-score evaluation run results can be seen for evaluation runs utilizing CNESS with all-tokens disabled and enabled respectively. These tables also list the hypothetical best F1-score evaluation for run the mean reference edge set evaluation runs. Additionally the mean standard deviation for these ERSs is shown, i.e. the mean of the standard deviations of the F1-score for every ERS of the form `semsim-ctx NM-AT r-N-* t-TS`.

Significant Observations

1.1 All evaluation runs utilizing NESS achieve a best F1-score that is higher than the F1-score of the random evaluation run and the original evaluation run

1.2 CNESS achieves achieves a higher F1-score than FNESS by 4.0%, when comparing the highest F1-scores achieved among all FNESS utilizing evaluation runs and the highest mean RES best F1 score achieved among all CNESS utilizing evaluation runs (`semsim-fix-lemma cn t-0.30` and `semxim-ctx e5 r-10-mean-best`)

1.3 Lemma based FNESS achieves a higher F1-score than non-lemma FNESS by 3.2%, when comparing the highest F1-scores achieved by evaluation runs utilizing one of the two variants (`semsim-fix w2v t-0.27` and `semsim-fix-lemma cn t-0.30`)

1.4 For lemma based FNESS, the conceptnet-numberbatch model achieves a higher best F1-score than the word2vec model by 4.2%
(`semsim-fix-lemma cn t-0.30` vs `semsim-fix-lemma w2v t-0.33`)

1.5 For not lemma based FNESS, the word2vec model achieves a higher best F1-score than the conceptnet-numberbatch model by 1.4%
(`semsim-fix w2v t-0.27` vs `semsim-fix cn t-0.25`)

Evaluation Run Name				Prec.	Rec.	(Best) F1-Score	
CP	NM	RES	t_s				Std. Dev.
random			-	0.300	0.500	**0.375**	-
original			-	0.706	0.209	**0.322**	-
semsim-fix	cn		0.25	0.479	0.524	0.500	-
semsim-fix	w2v		0.27	0.483	0.533	**0.507**	-
semsim-fix-l.	cn		0.30	0.492	0.558	**0.523**	-
semsim-fix-l.	w2v		0.33	0.460	0.553	0.502	-
semsim-ctx	e5	r-1-mean-best	0.65	0.392	0.772	0.518	+/- 0.025
semsim-ctx	gte	r-1-mean-best	0.59	0.336	0.879	0.483	+/- 0.025
semsim-ctx	e5	r-3-mean-best	0.68	0.399	0.818	0.536	+/- 0.021
semsim-ctx	gte	r-3-mean-best	0.65	0.365	0.799	0.499	+/- 0.016
semsim-ctx	e5	r-10-mean-best	0.72	0.416	0.790	**0.544**	+/- 0.020
semsim-ctx	gte	r-10-mean-best	0.68	0.382	0.812	0.517	+/- **0.010**
semsim-ctx	e5-at	r-1-mean-best	0.69	0.369	0.841	0.509	+/- 0.016
semsim-ctx	gte-at	r-1-mean-best	0.66	0.335	0.882	0.483	+/- 0.021
semsim-ctx	e5-at	r-3-mean-best	0.72	0.378	0.821	0.516	+/- 0.011
semsim-ctx	gte-at	r-3-mean-best	0.70	0.336	0.876	0.485	+/- 0.017
semsim-ctx	e5-at	r-10-mean-best	0.74	0.382	0.843	**0.525**	+/- 0.012
semsim-ctx	gte-at	r-10-mean-best	0.72	0.338	0.900	0.491	+/- **0.008**

Table 6.5: Evaluation scores corresponding to best F1-scores for all evaluation runs

1.6 CNESS with the AT option disabled achieves a higher or equal mean RES best F1-score than CNESS with AT option enabled in 6/6 (100%) direct comparisons (`semsim-ctx NM r-N-mean-best` vs `semsim-ctx NM-at r-N-mean-best`)

1.7 CNESS with the E5 model achieves a higher or equal mean RES best F1-score than CNESS with the GTE model in 6/6 (100%) direct comparisons (`semsim-ctx e5-* r-N-mean-best` vs `semsim-ctx gte-* r-N-mean-best`)

1.8 CNESS with the AT option enabled has a lower standard deviation of best F1-score than CNESS with the AT option disabled in 5/6 (83%) direct comparisons (`semsim-ctx NM-at r-N-mean-best` vs `semsim-ctx NM r-N-mean-best`)

6.4.2 Evaluation Metric vs. Similarity Threshold

These plots visualise the resulting evaluation scores for the different evaluation metrics in relation to different values for the similarity threshold.

Figure 6.2a shows the F1-score vs. the ST for the best performing evaluation runs for each conflict pattern. That means for every conflict pattern, this shows the evaluation run(s) with the configuration that resulted in the highest best F1-score. For the `random` and `original` evaluation runs, there is obviously no configuration to choose from.

For the FNESS utilizing evaluation runs, the evaluation run with the highest F1-score in the ERSs of the form `semsim-fix(-lemma) NM` is selected. This means for the semsim-fix and semxim-fix-lemma conflict patterns the corresponding best F1-score evaluation runs of the best performing model are selected..

For the CNESS utilizing evaluation runs, the evaluation runs which correspond to highest mean RES best F1-score are selected. This means the ERS of the form `semsim-ctx NM-AT r-N-*` which resulted in the highest mean value of best F1-scores. This ERS consists of five evaluation runs, wich each have the form `semsim-ctx NM-AT r-N-X`. The F1-scores for these runs are plotted with a lighter curve. The additional synthetical `semsim-ctx NM-AT r-N-mean` score is plotted with a normally bold curve.

Figure 6.2b follows the same concept as fig. 6.2a, but instead of the F1-score this plot shows the scores of precision and recall vs. the ST. Also in the selection of evaluation runs, the semsim-fix (not lemma) conflict pattern based evaluation runs are excluded.

The plots in this section are selected because they are deemed to be most relevant for the following. A more comprehensive comparison of the different evaluation runs from this perspective can be found in appendix A.4.

Active Similarity Threshold Range To facilitate the description of observation for this perspective on the result data, the concept of the *active similarity threshold range* (ASTR) is introduced. For a given ESR of the form `semsim-* NM(-AT) (r-N-X)`, the ASTR describes the range of t_s for which the recall (r) that is achieved by these evaluation runs is $r \neq r_{max} = 1$ and $r \neq r_{min}$. Here r_{min} and r_{max} are the lowest and highest recall values, which correspond to $t_{s1} <= t_{s2}$, since the function $r(t_s)$ is monotonically decreasing.

Significant Observations

2.1 The ASTR of `semsim-fix-lemma cn` is larger ($0.0 < t_s < 1.0$) than the ASTR of `semsim-ctx e5` (ca. $0.625 < t_s < 0.875$)

2.2 Generally the ASTR of ERSs of the form `semsim-fix-* NM` is larger than the the ASTR of ERSs of the form `semsim-ctx` (confer also fig. A.1, fig. A.2 fig. A.5 and fig. A.6)

2.3 The ASTRs are nearly equal for ERSs of the form `semsim-fix-* NM` (confer fig. A.1 and fig. A.2)

2.4 The ASTR of ERSs of the form `semsim-ctx gte-AT` begin at a lower value than for ERSs of the form `semsim-ctx e5-AT` (confer fig. A.5 and fig. A.4)

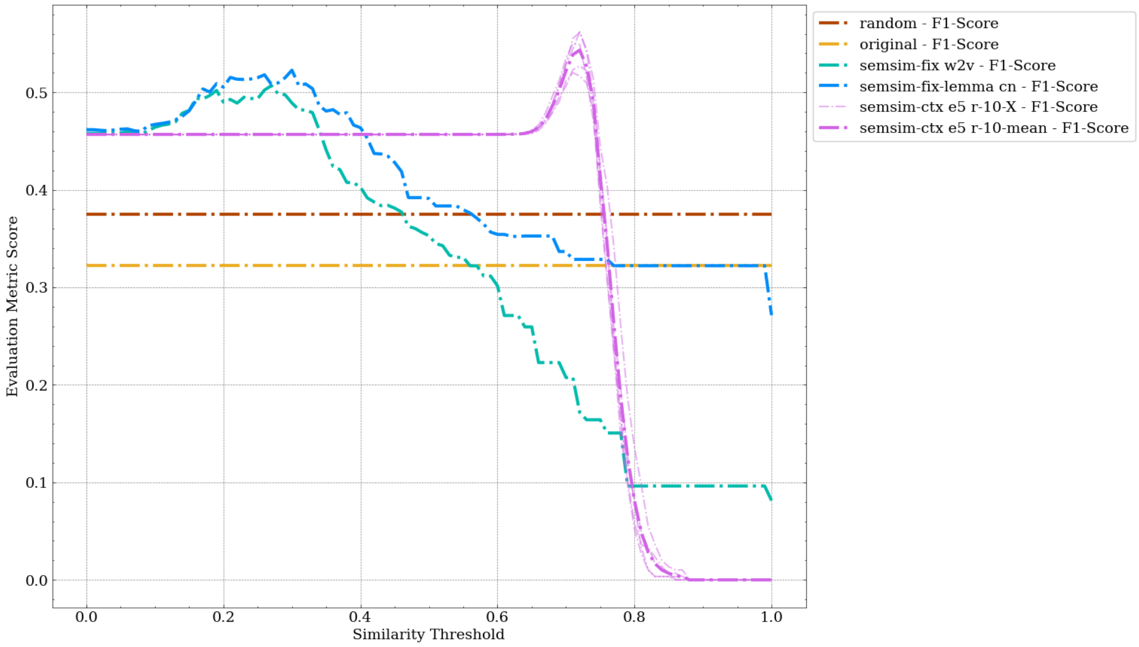
2.5 The ASTR of ERSs of the form `semsim-ctx NM` begin at a lower value than for ERSs of the form `semsim-ctx NM-at` (confer fig. A.6 and fig. A.4)

2.6 The ASTRs end at nearly the same value for ERSs of the form `semsim-ctx NM-AT` (confer fig. A.5, fig. A.6 and fig. A.4)

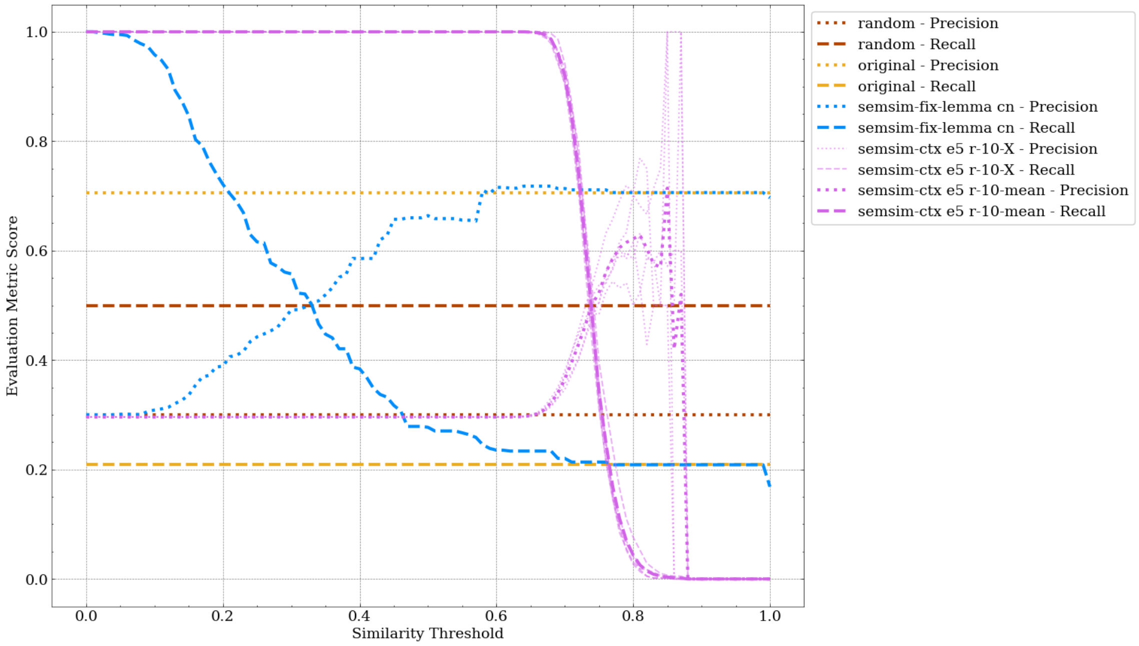
2.7 The evaluation runs in the ERS `semsim-fix-lemma cn` achieve a higher F1 value than the evaluation run in the ERS `semsim-fix w2v` for nearly all values of t_s

2.8 The precision of the evaluation run which achieves the highest precision among those in the synthetic ESR `semsim-ctx e5 r-10-mean` ($p_{\max}$) and the precision of the original evaluation run (p_{og}) are nearly equal ($p_{\max} \approx p_{\text{og}}$)

2.9 The precision of evaluation run `semsim-fix-lemma cn` correlates with the ST until it reaches the value achieved by the `original` evaluation run, where it plateaus



(a) F1-score vs. ST for the evaluation runs `random`, `original`, `semsim-fix w2v`, `semsim-fix-lemma cn` and `semsim-ctx e5 r-10-X`



(b) Precision and recall vs. ST for the evaluation runs `random`, `original`, `semsim-fix-lemma cn` and `semsim-ctx e5 r-10-X`

Figure 6.2: Evaluation metric scores vs. similarity threshold values

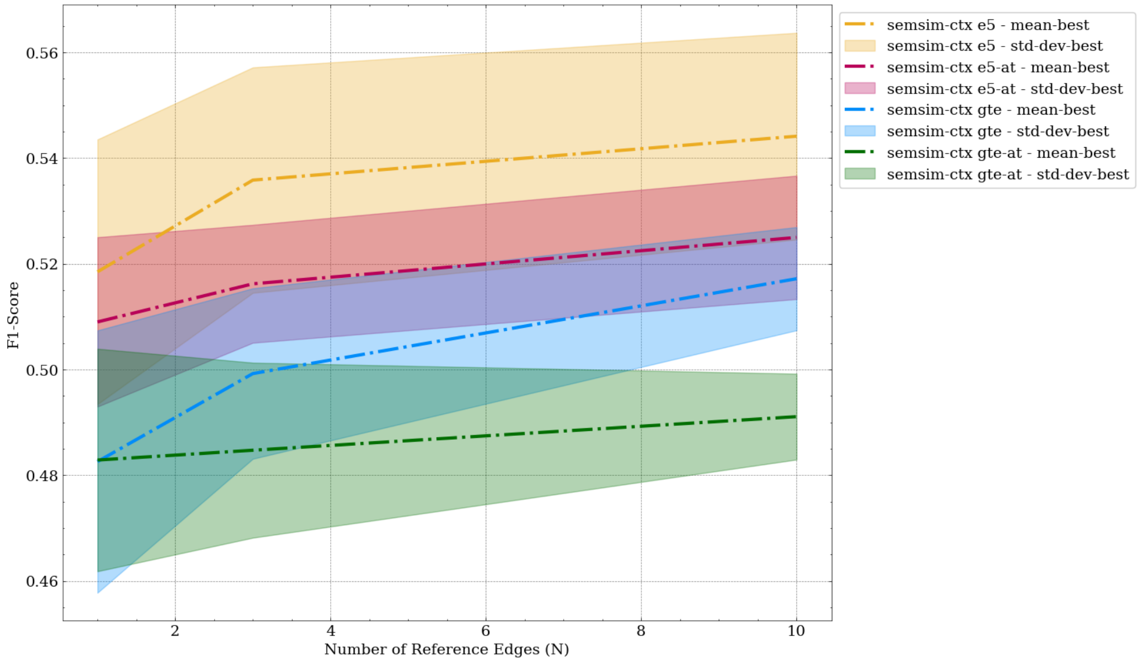


Figure 6.3: Mean reference edge set best F1-score for the ERSs `semsim-ctx e5 r-N-*`, `semsim-ctx e5-at r-N-*`, `semsim-ctx gte-at r-N-*` and `semsim-ctx gte-at r-N-*`

2.10 The precision of the evaluation runs in ERS `semsim-ctx e5` correlate with the ST until precision (p) and recall (r) reach approximately the same value ($p \approx r \approx 0.5$), after which it fluctuates (the specific fluctuation varies with the specific RES)

2.11 The precision achieved by evaluation runs in ERSs of the form `semsim-ctx e5 r-10-* t-TS` has a higher variation for $t_s \geq 0.75$ than for $t_s < 0.75$

2.12 The precision of the evaluation run which achieves the highest precision among those in the synthetic ESR `semsim-ctx e5 r-10-mean` (p_{max}) and the precision of the original evaluation run (p_{og}) are nearly equal ($p_{\text{max}} \approx p_{\text{og}}$)

2.13 The evaluation metric scores of `semsim-fix-lemma cn t-1.00` are lower than those of the `original` evaluation run

6.4.3 Best F1-Score vs. Number of Reference Edges

Figure 6.3 visualises the relation of the number of reference edges and the best-F1 score. For this purpose the number of reference edges `N` is plotted versus the mean RES best F1-score for all ERSs of the form `semsim-ctx NM-AT r-N-*`. The standard deviation of the best F1 scores for these ERSs is visualised by the shaded areas around the curves of the mean RES best F1-scores.

Significant Observations

3.1 The mean RES best F1-score of the evaluation runs in an ERS of the form `semsim-ctx NM(-AT) r-`N_1`-*` is higher than the mean RES best F1-score of the evaluation runs in an ERS of the form `semsim-ctx NM(-AT) r-`N_2`-*`, if $N_2 > N_1$ for $N_1, N_2 \in \{1, 3, 10\}$

3.2 The standard deviation of the mean RES best F1-score of the evaluation runs in an ERS of the form `semsim-ctx NM(-AT) r-`N_1`-*` is lower than the standard deviation of the mean RES best F1-score of the evaluation runs in an ERS of the form `semsim-ctx NM(-AT) r-`N_2`-*`, if $N_2 > N_1$ for $N_1, N_2 \in \{1, 3, 10\}$, except for `semsim-ctx et-at r-3-*` ($sd_1 = 0.11$) and `semsim-ctx et-at r-10-*` ($sd_2 = 0.12$)

6.4.4 Predicate Lemma based Evaluation Run Comparison

In this section it is explored how the different NESS systems differ in which edges they match. This is done by following up on the concept of the predicate lemma introduced in section 6.2.1. In section 6.2.2 one of the two desired characteristics of the dataset states that it should contain the largest possible number of edges per unique predicate lemma. Specifically it is of interest, how the CNESS system performs in comparison to the FNESS system for subsets of edges which share the same predicate lemma. Such a subset of edges of the conflict dataset is in the following referred to as *Predicate Lemma Edge Set* (PLES).

NESS Type Representatives Two evaluation runs are selected to represent the two versions of the NESS systems for the comparison. The lemma-based version of FNESS is chosen here, because of its superior performance regarding best F1-score that is observed in section 6.4.1 and section 6.4.2. Specifically, the evaluation runs `semsim-fix-lemma cn t-0.30` (evaluation run A) and `semsim-ctx e5 r-10-2 t-0.72` (evaluation run B) are selected as representatives of the FNESS and CNESS system respectively. These are the best performing evaluation runs regarding F1-score for the respective semsim conflict patterns (i.e. NESS system), which can be seen in table 6.5 and table A.6.[4]

PLES Evaluation Score based Evaluation Run Comparison

Label Balance Ratio The *Label Balance Ratio* (LBR) measures how balanced the labels in a given set of labelled edges are. It is calculated by eq. (6.1). Here n_{pos} and n_{neg} are the number of positively ("conflict") and negatively ("no conflict") labeled edges in the edge set. An edge set with fully balanced labels has $LBR = 1$ and completely unbalanced labeled edge set has $LBR = 0$.

$$LBR = 1 - \left(\frac{|n_{\text{pos}} - n_{\text{neg}}|}{n_{\text{pos}} + n_{\text{neg}}} \right) \tag{6.1}$$

[4] The latter table specifically shows that the evaluation runs `semsim-ctx e5 r-1-2 t-0.67`, `semsim-ctx e5 r-10-2 t-0.72` and `semsim-ctx e5 r-10-4 t-0.72` all correspond to an F1 score $s_{\text{F1}} = 0.56$. The evaluation runs utilizing the RESs of size $N = 10$ are chosen over the one utilizing an RES of size $N = 1$, because of their generally superior performance regarding F1 score, which is observed in section 6.4.3. Among the two remaining evaluation runs, `semsim-ctx e5 r-10-2 t-0.72` is selected randomly.

The evaluation metrics are computed for evaluation run A and B for each PLES, along with metrics that measure distributional properties of a PLES. Namely the number of edges n_e, the number of positively labeled and negatively labeled edges (n_{pos} and n_{neg}), the LBR and the entropy of a PLES.

Table 6.6 lists the ten predicate lemmas for whose PLES the absolute difference in F1-score, which is achieved in the two evaluation runs, is the highest. Conversely table 6.8 lists the ten predicate lemma for whose PLES the difference in F1-score which is achieved in the tow evaluation runs is the lowest. In both tables the predicate lemmas have been filtered beforehand, so that only PLES with $n_s >= 5$ samples are considered. Additionally the recalls (r_A, r_B) achieved by both evaluation runs regarding a PLES must fulfil the condition that $r_A + r_B > 0$, i.e. at least the recall achieved by one of the evaluation runs must be non-zero. Table 6.7 follows the same concept as table 6.6, except for the condition regarding the recalls being $r_A * r_B > 0$, i.e. the recall achieved by the both evaluation runs must be non-zero. This second variant of the table is not shown for table 6.8, because it only lists lemmas for whose PLES the F1-score achieved by both evaluation runs is zero.

Exemplary edges that correspond to the lemmas in table 6.6 and table 6.7 are shown in table 6.9 and table 6.10. They are presented in combination with the labels given in the dataset and those produced by both evaluation runs. Both tables contain two edges per lemma, with one edge labeled as "conflict" and the other as "no conflict". In table 6.9, evaluation run B labels all edges correctly. Conversely, in table 6.10, evaluation run B labels all edges incorrectly. Since evaluation run A can only produce the same label for every edge that corresponds to the same lemma, one edge per lemma is labeled correctly and one edge is labeled incorrectly by evaluation run A.

Significant Observations

4.1 The CNESS utilizing evaluation run achieves a higher F1-score than the FNESS utilizing evaluation run for every PLES of the the top ten PLES regarding highest difference in F1-score between the two evaluation runs (independently of the recall condition)

4.2 The mean LBR and mean entropy of the top ten PLESs regarding the highest difference in F1-score between the two evaluation runs are higher than the mean LBR and mean entropy of the top ten PLESs regarding the lowest difference in F1-score between the two evaluation runs (independently of the recall condition)

4.3 The differences in F1-score achieved by the two evaluation runs is higher for the recall condition $r_A + r_B > 0$ than for $r_A \cdot r_B > 0$, because for the first condition the F1-score of the FNESS evaluation run is zero for all PLESs

4.4 The lemmas corresponding to the the top ten PLES regarding highest difference in F1-score between the two evaluation runs are all not included in the conflict predicates: "accuse", "arrest", "clash", "condemn", "kill", "slam"

4.5 Of the lemmas corresponding to the top ten PLES regarding lowest difference in F1-score between the two evaluation runs, five of six are included in the conflict predicates: "arrest", "condemn", "kill", "slam" ("clash" is not included)

Lemma	F1 Diff.	F1 A	F1 B	n_e	$n_{\text{pos}}/n_{\text{neg}}$	LBR	Entropy
file	1.00	0.00	1.00	5	5/0	0.00	0.00
order	0.88	0.00	0.88	14	7/7	1.00	1.00
launch	0.86	0.00	0.86	14	8/6	0.86	0.99
step	0.80	0.00	0.80	5	2/3	0.80	0.97
target	0.77	0.00	0.77	8	6/2	0.50	0.81
use	0.75	0.00	0.75	14	5/9	0.71	0.94
block	0.73	0.00	0.73	8	4/4	1.00	1.00
take	0.71	0.00	0.71	19	6/13	0.63	0.90
open	0.67	0.00	0.67	8	1/7	0.25	0.54
suspend	0.67	0.00	0.67	6	2/4	0.67	0.92
build	0.67	0.00	0.67	6	1/5	0.33	0.65
						mean	
						0.61	0.79

Table 6.6: Top ten lemmas regarding the highest difference in F1-score between the evaluation runs with recalls $r_A + r_B > 0$ and number of samples per lemma $n_s >= 5$ for the evaluation runs `semsim-fix-lemma cn t-0.30` (A) and `semsim-ctx e5 r-10-2 t-0.72` (B)

Lemma	F1 Diff.	F1 A	F1 B	n_e	$n_{\text{pos}}/n_{\text{neg}}$	LBR	Entropy
accept	0.42	0.25	0.67	7	1/6	0.29	0.59
strike	0.30	0.80	0.50	9	6/3	0.67	0.92
capture	0.22	0.44	0.67	7	2/5	0.57	0.86
seize	0.17	0.67	0.50	12	6/6	1.00	1.00
deny	0.17	0.50	0.67	6	2/4	0.67	0.92
suggest	0.17	0.33	0.50	5	1/4	0.40	0.72
warn	0.12	0.93	0.81	24	21/3	0.25	0.54
claim	0.12	0.43	0.55	22	6/16	0.55	0.85
attack	0.09	0.91	1.00	12	10/2	0.33	0.65
threaten	0.09	0.52	0.61	20	7/13	0.70	0.93
approve	0.08	0.12	0.20	16	1/15	0.12	0.34
						mean	
						0.50	0.76

Table 6.7: Top ten lemmas regarding the highest difference in F1-score between the evaluation runs with recalls $r_A \cdot r_B > 0$ and number of samples per lemma $n_s >= 5$ for the evaluation runs `semsim-fix-lemma cn t-0.30` (A) and `semsim-ctx e5 r-10-2 t-0.72` (B). Demonstrating cases, where evaluation run B produces correct labels.

Lemma	F1 Diff.	F1 A	F1 B	n_e	$n_{\text{pos}}/n_{\text{neg}}$	LBR	Entropy
arrest	0.00	1.00	1.00	11	11/0	0.00	0.00
slam	0.00	0.92	0.92	7	6/1	0.29	0.59
criticize	0.00	0.91	0.91	6	5/1	0.33	0.65
shoot	0.00	0.89	0.89	5	4/1	0.40	0.72
condemn	0.00	0.84	0.84	25	18/7	0.56	0.86
dismiss	0.00	0.80	0.80	6	4/2	0.67	0.92
tell	0.01	0.49	0.50	28	9/19	0.64	0.91
accuse	0.02	0.97	0.95	33	31/2	0.12	0.33
kill	0.02	0.66	0.64	77	38/39	0.99	1.00
say	0.03	0.45	0.48	31	9/22	0.58	0.87
						mean	
						0.46	0.68

Table 6.8: Top ten lemmas regarding the lowest absolute difference in F1-score between the evaluation runs with recalls $r_A + r_B > 0$ and number of samples per lemma $n_s >= 5$ for the evaluation runs `semsim-fix-lemma cn t-0.30` (A) and `semsim-ctx e5 r-10-2 t-0.72` (B)

Lemma	Edge Content	Dataset	Eval. A	Eval. B
deny	Turkish FM denies investigation claims on German firms	no conflict	conflict	no conflict
	Kenya to deny entry to Ebola states	conflict	conflict	conflict
claim	Bangkok police claim reward in Erawan Shrine bomber hunt	no conflict	conflict	no conflict
	Isis claims responsibility for killing of Hindu priest in Bangladesh	conflict	conflict	conflict
launch	ISIS launches assault on pro-Assad forces in western Syria	conflict	no conflict	conflict
	UK PM Cameron launches English language campaign for Muslim women	no conflict	no conflict	no conflict
step	Beijing Steps Up Crackdown on Hong Kong Lawmakers	conflict	no conflict	conflict
	Irish Prime Minister Enda Kenny to step down	no conflict	no conflict	no conflict

Table 6.9: Dataset labels and evaluation labels of exemplary edges for evaluation runs `semsim-fix-lemma cn t-0.30` (A) and `semsim-ctx e5 r-10-2 t-0.72` (B). Demonstrates cases, in which evaluation run B produces correct labels.

Lemma	Edge Content	Dataset	Eval. A	Eval. B
take	Japanese porn industry takes first step toward recognizing it has a problem	no conflict	no conflict	conflict
	How flash mob flamenco took on the banks	conflict	no conflict	no conflict
deny	Indian government denies renewal of passport opposition lawyer over minor traffic rule violation	conflict	conflict	no conflict
	Russia denies ground troop in Syria	no conflict	conflict	conflict
strike	UK strikes first ISIS targets	conflict	conflict	no conflict
	Magnitude 6.9 quake strikes Sichuan region of China	no conflict	conflict	conflict
say	Denmark Says Microsoft Owes £660 Million in Tax	conflict	conflict	no conflict
	China Says Its Working with Latin America for a New World Order	no conflict	conflict	conflict

Table 6.10: Dataset labels and evaluation labels of exemplary edges for evaluation runs `semsim-fix-lemma cn t-0.30` (A) and `semsim-ctx e5 r-10-2 t-0.72` (B). Demonstrates cases, in which evaluation run B produces incorrect labels.

6.5 Result Discussion

In this section the previously presented evaluation results are discussed. It synthesises the major insights derived from the observations of the different result data perspectives. The discussion is organised into categories and sub-categories, which relate to the research questions outlined in section 3.1. Here retrieval performance generally refers to joined measure of precision and recall and therefore means F1-score, as stated above in section 6.3.2.

Evaluation Limitations

Prior to the result synthesis, we discuss some general limitations of this evaluation:

Case Study A central limitation of this evaluation is its reliance on a single case study. Consequently, it is unclear whether the insights pertaining to the behaviour of NESS-SHPM can be generalised across different applications and datasets, particularly regarding the optimal values for the similarity threshold.

Dataset Another limitation of this evaluation lies in the dataset. Specifically its relatively small size of $n_s = 2000$ samples and the missing cross-validation of annotation labels.

NESS Model Selection The evaluation is also limited to the set of NESS models, that was integrated into the NESS-SHPM implementation. However, these models have been purposefully selected based on a.o. their performance (see section 2.3). We argue therefore that they serve as suitable representatives of their respective NESS type.

Retrieval Performance Metric In this evaluation, F1-Score is chosen as overall retrieval performance metric, since it is an established metric for text retrieval and classification tasks and closely relates to precision and recall. There exists other metrics, such as the *Matthews Correlation Coefficient* (MCC), which can be used to assess retrieval or classification performance. The MMC has recently been identified to be more suitable for many binary classification tasks (Chicco and Jurman 2020), which is a possible framing for the task evaluated here. Nonetheless, we use F1-Score because it is very well established and relates closely to precision and recall, specifically sharing the same value range. This makes it especially useful for jointly plotting these metrics. However, it shall be noted that the F1-Score of the original evaluation run is lower than of the random evaluation run, which indicates a legitimate interest for using another metric to assess retrieval (or classification) performance in this case study. An MCC plot for the evaluation runs `random`, `original`, `semsim-fix w2v`, `semsim-fix-lemma cn` and `semsim-ctx e5 r-10-X` is given in fig. A.7 in appendix A.4. Here the random evaluation run shows lower retrieval performance than the original evaluation run. The relation of the retrieval performance of NESS-SHPM and symbolic matching (i.e. the original conflict pattern) does not change in principle. Though, the effect of NESS type and NESS configurations on retrieval performance of NESS-SHPM differs in comparison to an F1-Score based evaluation.

Result Synthesis

The following section consolidates the primary insights obtained from examining the signification observations across the different result data perspectives.

6.5.1 Retrieval Performance Improvement

- NESS-SHPM can achieve a better retrieval performance than the original conflict pattern, independent of NESS type and configuration (depending on the ST)
  Supporting Observations: Item 1.1
- CNESS-SHPM utilising sub-edge specific embeddings can achieve the overall best retrieval performance (in comparison to FNESS-SHPM and the original pattern)
  Supporting Observations: Item 1.1, Item 1.2
- Using lemma-based FNESS-SHPM instead of word-based FNESS-SHPM can achieve a better retrieval performance
  Supporting Observations: Item 1.3, Item 2.7

Similarity Threshold Impact

- The relation of ST and NESS-SHPM retrieval performance depends primarily on the NESS type (given the neural embedding models selected in this work)
  Supporting Observations: Item 2.1 Item 2.2 Item 2.3
- The relation of ST and CNESS-SHPM retrieval performance depends secondarily on the NESS model and the usage of the all tokens option
  Supporting Observations: Item 2.4 Item 2.5 Item 2.6

NESS Configuration Impact

- Using a generally better performing NESS model (regarding established benchmarks) does not generally improve the NESS-SHPM retrieval performance
Supporting Observations: Item 1.4, Item 1.5, Item 1.7

- Using the sub tokens embedding instead of the all tokens embedding improves CNESS-SHPM retrieval performance, but using the all tokens embedding makes it less sensible to the selection of the reference edges
Supporting Observations: Item 1.8, Item 1.8

- CNESS-SHPM retrieval performance improves with a higher number of reference edges and is less sensible to the specific selection of reference edges
Supporting Observations: Item 3.1, Item 3.2

6.5.2 Retrieval Precision Behaviour

- The precision of NESS-SHPMM correlates with the ST until a specific value of the ST, which itself is specific to the NESS type, NESS model (and other CNESS parameters, especially the selection of reference edges)
Supporting Observations: Item 2.9, Item 2.10, Item 2.13

- CNESS-SHPM achieves on average the same precision as the original conflict pattern and lemma-based FNESS, although CNESS-SHPM can achieve a higher precision dependent on the selection of the reference edges
Supporting Observations: Item 2.11, Item 2.12

6.5.3 Contextual Differentiation Ability

- CNESS-SHPM is able differentiate when matching against a set of edges where purely symbolic SHPM and FNESS-SHPMM cannot, i.e. in cases where context is needed to determine the correct semantics of a word
Supporting Observations: Item 4.1, Item 4.2, Item 4.4, Item 4.5

- While CNESS-SHPM achieves a highest difference in retrieval performance for sets of edges, where FNESS-SHPM does not match, it also achieves a better retrieval performance in cases where FNESS-SHPM does match
Supporting Observations: Item 4.1 Item 4.3

7 Conclusion

This work advances the research on text analysis systems, specifically aimed at Computational Social Science researchers. It contributes the capability of content generalisation to the Semantic Hypergraph framework's pattern matching by integrating neural embedding-based semantic similarity measurement. Thereby enabling adaptivity of the SH pattern matching, while providing partial control over the simultaneously introduced opaqueness, through the configuration of the similarity threshold, the ability to choose between fixed word and contextual neural embedding-based semantic similarity, and the option to toggle sub-edge specificity of the contextual embeddings. Additionally the effectiveness of NESS extended pattern matching in regard to its retrieval performance is shown in a quantitative evaluation, specifically showcasing the systems ability for contextual differentiation.

In correspondence to the specific research questions posed in section 3.1, we presented the following specific contributions of this work:

C.1 Conceived and implemented neural embedding-based semantic similarity extended Semantic Hypergraph pattern matching, which includes extending the pattern language, modifying the pattern matching process and devising a procedure for constructing sub-edge specific contextual embeddings. These innovations for the SH framework were implemented in a readily usable manner within the publicly available graphbrain software package.

C.2 Identified fixed word and contextual neural embedding models that are suited to assess semantic similarity in the SH matching process by conducting a literature review and examining state-of-the-art models in regards to their implementation intricacies.

C.3 Constructed a labeled dataset of hyperedges building upon previous work on conflict expressions, which can be utilised to quantitatively evaluate the retrieval performance of the SH pattern matching.

C.4 Showed that NESS extended pattern matching can produce better retrieval performance than purely symbolic matching, if the similarity threshold is chosen accordingly. Specifically, the utilisation of sub-edge specific contextual embeddings is shown to be able to produce the best overall retrieval performance. Particularly showed that, a higher retrieval precision than through purely symbolic pattern matching can be achieved by CNESS-SHPM, although this depends strongly on the selection of reference edges.

C.5 Exhibited that contextual NESS extended pattern matching is in principle able to differentiate between desired and undesired edges in cases where fixed word NESS cannot.

C.6 Demonstrated that measuring contextual NESS utilizing sub-edge specific contextual embeddings produces better retrieval performance than by utilizing contextual embeddings of the entire edge content. While both methods exhibit improved retrieval performance with an increased number of reference edges, the performance of the latter approach is less sensitive to the choice of reference edges.

C.7 Indicated that that retrieval precision and therefore content generalisation of NESS-SHPM correlates with the controllable similarity threshold for certain a value range. This range depends on primarily on the NESS type, given the neural embedding model selection in this work. As expected, retrieval recall monotonically decreases inversely with the similarity threshold value.

C.8 Found that the choice of neural embeddings models has an impact on NESS-SHPM retrieval performance. However, in the limited model selection of this work, model performance on established semantic similarity benchmarks does not align with NESS-SHPM retrieval performance.

Future Work

Possible future work, building upon this work, may involve the following research areas:

Similarity Threshold Control

For practical applications, choosing a similarity threshold that corresponds to the desired semantic content generalisation level is crucial. The similarity threshold in the default configurations of graphbrain for both SemSim matcher types has been set to the threshold values, which resulted in the best retrieval performance when evaluating on the conflict dataset. However, it is uncertain how well this generalises to other applications.

A possibility to efficiently identify the relevant similarity threshold range and most suitable values for a given application could enhance the practical usability of NESS-SHPM.

Contextual Embedding Construction

Exploring the adaptation of contextual embedding construction and its impact on NESS-SHPM appears promising. Specifically, we propose two research directions in this regard:

On the one hand, allowing more fine-grained control over the specificity of the contextual embeddings. By that, we mean the possibility to configure not only, if the embeddings are constructed by averaging the sub-edge specific or all token embeddings of the reference edge. Instead it is conceivable to allow for a specific degree of sub-edges upwards the hyperedge hierarchy to be included in the embedding construction. Another approach could be to use the token embeddings corresponding to the sub-edges captured by multiple variables for the construction of the final embedding.

On the other hand, the level contextuality of the embeddings produced by contextual embedding models could be influenced. Following the methodology on investigating embedding contextuality outlined by Ethayarajh 2019, this could be achieved by using the states of different hidden layers of the model, rather than exclusively the last layer, to construct the embeddings. This approach could also serve as an alternative method for producing fixed word embeddings (Gupta and Jaggi 2021).

Implementation Improvements

We assess that the implementation of NESS-SHPM would primarily benefit in terms of efficiency through improvements to the data structure managing the constructed contextual embeddings. Rather than caching the embeddings, they could be stored in a vector database, which could then also directly perform the cosine similarity computations.

Further Evaluations

The retrieval performance and other behaviours of NESS-SHPM could be evaluated using additional datasets. Specifically, this evaluation could test whether the similarity threshold values that yielded the best retrieval performance on the conflict dataset are generalisable to other applications.

Acronyms

AT All Tokens

BES Base Edge Set

CBSS Corpus-based Semantic Similarity

CD Conflict Dataset

CNESS Contextual Neural Embedding-based Semantic Similarity

CNESS-SHPM Contextual Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching

CSS Computational Social Science

DL Deep Learning

DLSS Deep Learning-based Semantic Similarity

DNN Deep Neural Network

ERS Evaluation Run Set

FBES Filtered Base Edge Set

FNESS Fixed Word Neural Embedding-based Semantic Similarity

FNESS-SHPM Fixed Word Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching

IE Information Extraction

IR Information Retrieval

KBSS Knowledge-based Semantic Similarity

LBR Label Balance Ratio

LFNESS Lemma-based Fixed Word Neural Embedding-based Semantic Similarity

LFNESS-SHPM Lemma-based Fixed Word Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching

ML Machine Learning

NESS Neural Embedding-based Semantic Similarity

NESS-SHPM Neural Embedding-based Semantic Similarity extended Semantic Hypergraph Pattern Matching

NL Natural Language

NLP Natural Language Processing

NNLM Neural (Network) Language Modelling

PLES Predicate Lemma Edge Set

RES Reference Edge Set

SH Semantic Hypergraph

SHPL Semantic Hypergraph Pattern Language

SHPM Semantic Hypergraph Pattern Matching

SR Semantic Relatedness

SS Semantic Similarity

ST Similarity Threshold

Bibliography

Aizawa, Akiko (Jan. 1, 2003). "An Information-Theoretic Perspective of Tf–Idf Measures". In: *Information Processing & Management* 39.1, pp. 45–65. ISSN: 0306-4573. DOI: 10.1016/S0306-4573(02)00021-3.

Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi (Aug. 2017). "Understanding of a Convolutional Neural Network". In: *2017 International Conference on Engineering and Technology (ICET).* 2017 International Conference on Engineering and Technology (ICET), pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

Almeida, Felipe and Geraldo Xexéo (May 1, 2023). *Word Embeddings: A Survey.* DOI: 10.48550/arXiv.1901.09069. arXiv: 1901.09069 [cs, stat]. URL: http://arxiv.org/abs/1901.09069. preprint.

Auer, Sören et al. (2007). "DBpedia: A Nucleus for a Web of Open Data". In: *The Semantic Web.* Ed. by Karl Aberer et al. Berlin, Heidelberg: Springer, pp. 722–735. ISBN: 978-3-540-76298-0. DOI: 10.1007/978-3-540-76298-0_52.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (May 19, 2016). *Neural Machine Translation by Jointly Learning to Align and Translate.* DOI: 10.48550/arXiv.1409.0473. arXiv: 1409.0473 [cs, stat]. URL: http://arxiv.org/abs/1409.0473. preprint.

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent (2000). "A Neural Probabilistic Language Model". In: *Advances in Neural Information Processing Systems.* Vol. 13. MIT Press.

Bojanowski, Piotr et al. (2017). "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146. DOI: 10.1162/tacl_a_00051.

Bollegala, Danushka and James O'Neill (Apr. 1, 2022). *A Survey on Word Meta-Embedding Learning.* arXiv e-prints. DOI: 10.48550/arXiv.2204.11660. URL: https://ui.adsabs.harvard.edu/abs/2022arXiv220411660B. preprint.

Chandrasekaran, Dhivya and Vijay Mago (Feb. 18, 2021). "Evolution of Semantic Similarity—A Survey". In: *ACM Computing Surveys* 54.2, 41:1–41:37. ISSN: 0360-0300. DOI: 10.1145/3440755.

Chen, Stanley F. and Joshua Goodman (Oct. 1, 1999). "An Empirical Study of Smoothing Techniques for Language Modeling". In: *Computer Speech & Language* 13.4, pp. 359–394. ISSN: 0885-2308. DOI: 10.1006/csla.1999.0128.

Chicco, Davide and Giuseppe Jurman (Jan. 2, 2020). "The Advantages of the Matthews Correlation Coefficient (MCC) over F1 Score and Accuracy in Binary Classification Evaluation". In: *BMC Genomics* 21.1, p. 6. ISSN: 1471-2164. DOI: 10.1186/s12864-019-6413-7.

Cho, Kyunghyun et al. (Oct. 2014). "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation.* SSST 2014. Ed. by Dekai Wu et al. Doha, Qatar: Association for Computational Linguistics, pp. 103–111. DOI: 10.3115/v1/W14-4012.

Chowdhary, K. R. (2020). “Natural Language Processing”. In: *Fundamentals of Artificial Intelligence*. Ed. by K.R. Chowdhary. New Delhi: Springer India, pp. 603–649. ISBN: 978-81-322-3972-7. DOI: 10.1007/978-81-322-3972-7_19.

Church, Kenneth Ward and Patrick Hanks (June 1989). “Word Association Norms, Mutual Information, and Lexicography”. In: *27th Annual Meeting of the Association for Computational Linguistics*. ACL 1989. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 76–83. DOI: 10.3115/981623.981633.

Collobert, Ronan and Jason Weston (July 5, 2008). “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”. In: *Proceedings of the 25th International Conference on Machine Learning*. ICML ’08. New York, NY, USA: Association for Computing Machinery, pp. 160–167. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390177.

Dai, Andrew M. and Quoc V. Le (Nov. 4, 2015). *Semi-Supervised Sequence Learning*. DOI: 10.48550/arXiv.1511.01432. arXiv: 1511.01432 [cs]. URL: http://arxiv.org/abs/1511.01432. preprint.

Deerwester, Scott et al. (1990). “Indexing by Latent Semantic Analysis”. In: *Journal of the American Society for Information Science* 41.6, pp. 391–407. ISSN: 1097-4571. DOI: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.

Devlin, Jacob et al. (June 2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. NAACL-HLT 2019. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

Ethayarajh, Kawin (Nov. 2019). “How Contextual Are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. EMNLP-IJCNLP 2019. Hong Kong, China: Association for Computational Linguistics, pp. 55–65. DOI: 10.18653/v1/D19-1006.

Evans, James A. and Pedro Aceves (July 1, 2016). *Machine Translation: Mining Text for Social Theory*. DOI: 10.1146/annurev-soc-081715-074206. URL: https://papers.ssrn.com/abstract=2822747. preprint.

Gabrilovich, Evgeniy, Shaul Markovitch, et al. (2007). “Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis.” In: *IJcAI*. Vol. 7, pp. 1606–1611.

Galassi, Andrea, Marco Lippi, and Paolo Torroni (Oct. 2021). “Attention in Natural Language Processing”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.10, pp. 4291–4308. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2020.3019893.

Ganitkevitch, Juri, Benjamin Van Durme, and Chris Callison-Burch (June 2013). “PPDB: The Paraphrase Database”. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. NAACL-HLT 2013. Ed. by Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff. Atlanta, Georgia: Association for Computational Linguistics, pp. 758–764.

Gao, Tianyu, Xingcheng Yao, and Danqi Chen (Nov. 2021). “SimCSE: Simple Contrastive Learning of Sentence Embeddings”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2021. Ed. by Marie-Francine Moens et al. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 6894–6910. DOI: 10.18653/v1/2021.emnlp-main.552.

García-Ferrero, Iker, Rodrigo Agerri, and German Rigau (Nov. 2021). “Benchmarking Meta-embeddings: What Works and What Does Not”. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Findings 2021. Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 3957–3972. DOI: 10.18653/v1/2021.findings-emnlp.333.

Goikoetxea, Josu, Aitor Soroa, and Eneko Agirre (June 15, 2018). “Bilingual Embeddings with Random Walks over Multilingual Wordnets”. In: *Knowledge-Based Systems* 150, pp. 218–230. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2018.03.017.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (Nov. 10, 2016). *Deep Learning*. MIT Press. 801 pp. ISBN: 978-0-262-33737-3. Google Books: omivDQAAQBAJ.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (Mar. 22, 2013). *Speech Recognition with Deep Recurrent Neural Networks*. DOI: 10.48550/arXiv.1303.5778. arXiv: 1303.5778 [cs]. URL: http://arxiv.org/abs/1303.5778. preprint.

Gupta, Prakhar and Martin Jaggi (Aug. 2021). “Obtaining Better Static Word Embeddings Using Contextual Embedding Models”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. ACL-IJCNLP 2021. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, pp. 5241–5253. DOI: 10.18653/v1/2021.acl-long.408.

Han, Mengting et al. (2021). “A Survey on the Techniques, Applications, and Performance of Short Text Semantic Similarity”. In: *Concurrency and Computation: Practice and Experience* 33.5, e5971. ISSN: 1532-0634. DOI: 10.1002/cpe.5971.

Harispe, Sébastien et al. (2015). *Semantic Similarity from Natural Language and Ontology Analysis*. DOI: 10.2200/S00639ED1V01Y201504HLT027. arXiv: 1704.05295 [cs].

Harris, Zellig S. (Aug. 1, 1954). “Distributional Structure”. In: *WORD* 10.2-3, pp. 146–162. ISSN: 0043-7956. DOI: 10.1080/00437956.1954.11659520.

He, Hua and Jimmy Lin (June 2016). “Pairwise Word Interaction Modeling with Deep Neural Networks for Semantic Similarity Measurement”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. NAACL-HLT 2016. Ed. by Kevin Knight, Ani Nenkova, and Owen Rambow. San Diego, California: Association for Computational Linguistics, pp. 937–948. DOI: 10.18653/v1/N16-1108.

Hirschberg, Julia and Christopher D. Manning (July 17, 2015). “Advances in Natural Language Processing”. In: *Science* 349.6245, pp. 261–266. DOI: 10.1126/science.aaa8685.

Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 1997). “Long Short-Term Memory”. In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

Honnibal, Matthew et al. (2020). “spaCy: Industrial-strength Natural Language Processing in Python”. In: DOI: 10.5281/zenodo.1212303.

Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom (June 2014). “A Convolutional Neural Network for Modelling Sentences”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. ACL 2014. Ed. by Kristina Toutanova and Hua Wu. Baltimore, Maryland: Association for Computational Linguistics, pp. 655–665. DOI: 10.3115/v1/P14-1062.

Kowsari, Kamran et al. (Apr. 2019). “Text Classification Algorithms: A Survey”. In: *Information* 10.4 (4), p. 150. ISSN: 2078-2489. DOI: 10.3390/info10040150.

Landauer, Thomas K. and Susan T. Dumais (1997). “A Solution to Plato’s Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge". In: *Psychological Review* 104.2, pp. 211–240. ISSN: 1939-1471. DOI: 10.1037/0033-295X.104.2.211.

Landauer, Thomas K, Peter W. Foltz, and Darrell Laham (Jan. 1, 1998). "An Introduction to Latent Semantic Analysis". In: *Discourse Processes* 25.2-3, pp. 259–284. ISSN: 0163-853X. DOI: 10.1080/01638539809545028.

Lazer, David et al. (Feb. 6, 2009). "Computational Social Science". In: *Science* 323.5915, pp. 721–723. DOI: 10.1126/science.1167742.

Lee, Daniel D. and H. Sebastian Seung (Oct. 1999). "Learning the Parts of Objects by Non-Negative Matrix Factorization". In: *Nature* 401.6755, pp. 788–791. ISSN: 1476-4687. DOI: 10.1038/44565.

Levy, Omer and Yoav Goldberg (2014). "Neural Word Embedding as Implicit Matrix Factorization". In: *Advances in Neural Information Processing Systems.* Vol. 27. Curran Associates, Inc.

Li, Bohan et al. (Nov. 2020). "On the Sentence Embeddings from Pre-trained Language Models". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).* EMNLP 2020. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, pp. 9119–9130. DOI: 10.18653/v1/2020.emnlp-main.733.

Li, Yang and Tao Yang (2018). "Word Embedding for Understanding Natural Language: A Survey". In: *Guide to Big Data Applications.* Ed. by S. Srinivasan. Studies in Big Data. Cham: Springer International Publishing, pp. 83–104. ISBN: 978-3-319-53817-4. DOI: 10.1007/978-3-319-53817-4_4.

Li, Zehan et al. (Aug. 6, 2023). *Towards General Text Embeddings with Multi-stage Contrastive Learning.* DOI: 10.48550/arXiv.2308.03281. arXiv: 2308.03281 [cs]. URL: http://arxiv.org/abs/2308.03281. preprint.

Liu, Qi, Matt J. Kusner, and Phil Blunsom (Apr. 13, 2020). *A Survey on Contextual Embeddings.* DOI: 10.48550/arXiv.2003.07278. arXiv: 2003.07278 [cs]. URL: http://arxiv.org/abs/2003.07278. preprint.

Liu, Yinhan et al. (July 26, 2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach.* DOI: 10.48550/arXiv.1907.11692. arXiv: 1907.11692 [cs]. URL: http://arxiv.org/abs/1907.11692. preprint.

Lopez-Gazpio, I. et al. (Oct. 15, 2019). "Word N-Gram Attention Models for Sentence Similarity and Inference". In: *Expert Systems with Applications* 132, pp. 1–11. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2019.04.054.

Lund, Kevin and Curt Burgess (1996). "Producing High-Dimensional Semantic Spaces from Lexical Co-Occurrence". In: *Behavior Research Methods, Instruments & Computers* 28.2, pp. 203–208. ISSN: 0743-3808. DOI: 10.3758/BF03204766.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (July 7, 2008). *Introduction to Information Retrieval.* Higher Education from Cambridge University Press. DOI: 10.1017/CBO9780511809071. URL: https://www.cambridge.org/highereducation/books/introduction-to-information-retrieval/669D108D20F556C5C30957D63B5AB65C.

McCann, Bryan et al. (June 20, 2018). *Learned in Translation: Contextualized Word Vectors.* DOI: 10.48550/arXiv.1708.00107. arXiv: 1708.00107 [cs]. URL: http://arxiv.org/abs/1708.00107. preprint.

Melamud, Oren, Jacob Goldberger, and Ido Dagan (Aug. 2016). "Context2vec: Learning Generic Context Embedding with Bidirectional LSTM". In: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning.* CoNLL 2016. Ed. by Stefan Riezler and Yoav Goldberg. Berlin, Germany: Association for Computational Linguistics, pp. 51–61. DOI: 10.18653/v1/K16-1006.

Menezes, Telmo and Camille Roth (Feb. 18, 2021). *Semantic Hypergraphs.* DOI: 10.48550/arXiv.1908.10784. arXiv: 1908.10784 [cs]. URL: http://arxiv.org/abs/1908.10784. preprint.

Mihalcea, Rada, Courtney Corley, and Carlo Strapparava (July 16, 2006). "Corpus-Based and Knowledge-Based Measures of Text Semantic Similarity". In: *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1.* AAAI'06. Boston, Massachusetts: AAAI Press, pp. 775–780. ISBN: 978-1-57735-281-5.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (June 2013). "Linguistic Regularities in Continuous Space Word Representations". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* NAACL-HLT 2013. Atlanta, Georgia: Association for Computational Linguistics, pp. 746–751.

Mikolov, Tomas et al. (Oct. 16, 2013a). *Distributed Representations of Words and Phrases and Their Compositionality.* DOI: 10.48550/arXiv.1310.4546. arXiv: 1310.4546 [cs, stat]. URL: http://arxiv.org/abs/1310.4546. preprint.

Mikolov, Tomas et al. (Sept. 6, 2013b). *Efficient Estimation of Word Representations in Vector Space.* DOI: 10.48550/arXiv.1301.3781. arXiv: 1301.3781 [cs]. URL: http://arxiv.org/abs/1301.3781. preprint.

Mikolov, Tomáš et al. (2010). "Recurrent Neural Network Based Language Model". In: Proc. Interspeech 2010, pp. 1045–1048. DOI: 10.21437/Interspeech.2010-343.

Miller, George A. (Nov. 1, 1995). "WordNet: A Lexical Database for English". In: *Communications of the ACM* 38.11, pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748.

Min, Bonan et al. (Sept. 14, 2023). "Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey". In: *ACM Computing Surveys* 56.2, 30:1–30:40. ISSN: 0360-0300. DOI: 10.1145/3605943.

Minaee, Shervin et al. (Apr. 17, 2021). "Deep Learning–based Text Classification: A Comprehensive Review". In: *ACM Computing Surveys* 54.3, 62:1–62:40. ISSN: 0360-0300. DOI: 10.1145/3439726.

Mohammad, Saif M. and Graeme Hirst (Mar. 8, 2012a). *Distributional Measures as Proxies for Semantic Relatedness.* DOI: 10.48550/arXiv.1203.1889. arXiv: 1203.1889 [cs]. URL: http://arxiv.org/abs/1203.1889. preprint.

– (Mar. 8, 2012b). *Distributional Measures of Semantic Distance: A Survey.* DOI: 10.48550/arXiv.1203.1858. arXiv: 1203.1858 [cs]. URL: http://arxiv.org/abs/1203.1858. preprint.

Muennighoff, Niklas et al. (Mar. 19, 2023). *MTEB: Massive Text Embedding Benchmark.* DOI: 10.48550/arXiv.2210.07316. arXiv: 2210.07316 [cs]. URL: http://arxiv.org/abs/2210.07316. preprint.

Niwa, Yoshiki and Yoshihiko Nitta (Aug. 1994). "Co-Occurrence Vectors From Corpora vs. Distance Vectors From Dictionaries". In: *COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics.* COLING 1994. Kyoto, Japan.

P., Sunilkumar and Athira P. Shaji (Dec. 2019). "A Survey on Semantic Similarity". In: *2019 International Conference on Advances in Computing, Communication and Control (ICAC3).* 2019 International Conference on Advances in Computing, Communication and Control (ICAC3), pp. 1–8. DOI: 10.1109/ICAC347590.2019.9036843.

Parsons, Van L. (2017). "Stratified Sampling". In: *Wiley StatsRef: Statistics Reference Online.* John Wiley & Sons, Ltd, pp. 1–11. ISBN: 978-1-118-44511-2. DOI: 10.1002/9781118445112.stat05999.pub2.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (Oct. 2014). "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empiri-*

cal Methods in Natural Language Processing (EMNLP). EMNLP 2014. Doha, Qatar: Association for Computational Linguistics, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

Peters, Matthew E. et al. (June 2018). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. NAACL-HLT 2018. New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237. DOI: 10.18653/v1/N18-1202.

Qiu, XiPeng et al. (Oct. 1, 2020). "Pre-Trained Models for Natural Language Processing: A Survey". In: *Science China Technological Sciences* 63.10, pp. 1872–1897. ISSN: 1869-1900. DOI: 10.1007/s11431-020-1647-3.

Reimers, Nils and Iryna Gurevych (Nov. 2019). "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. EMNLP-IJCNLP 2019. Ed. by Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, pp. 3982–3992. DOI: 10.18653/v1/D19-1410.

Rohde, Douglas LT, Laura M Gonnerman, and David C Plaut (2006). "An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence". In: *Communications of the ACM* 8.627-633, p. 116.

Rudin, Cynthia (May 2019). "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead". In: *Nature Machine Intelligence* 1.5 (5), pp. 206–215. ISSN: 2522-5839. DOI: 10.1038/s42256-019-0048-x.

Salle, Alexandre, Marco Idiart, and Aline Villavicencio (June 3, 2016). *Enhancing the LexVec Distributed Word Representation Model Using Positional Contexts and External Memory*. DOI: 10.48550/arXiv.1606.01283. arXiv: 1606.01283 [cs]. URL: http://arxiv.org/abs/1606.01283. preprint.

Salle, Alexandre and Aline Villavicencio (June 2018). "Incorporating Subword Information into Matrix Factorization Word Embeddings". In: *Proceedings of the Second Workshop on Subword/Character LEvel Models*. SCLeM 2018. New Orleans: Association for Computational Linguistics, pp. 66–71. DOI: 10.18653/v1/W18-1209.

– (Aug. 19, 2019). *Why So Down? The Role of Negative (and Positive) Pointwise Mutual Information in Distributional Semantics*. DOI: 10.48550/arXiv.1908.06941. arXiv: 1908.06941 [cs]. URL: http://arxiv.org/abs/1908.06941. preprint.

Salle, Alexandre, Aline Villavicencio, and Marco Idiart (Aug. 2016). "Matrix Factorization Using Window Sampling and Negative Sampling for Improved Word Representations". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. ACL 2016. Berlin, Germany: Association for Computational Linguistics, pp. 419–424. DOI: 10.18653/v1/P16-2068.

Sezerer, Erhan and Selma Tekir (Oct. 4, 2021). *A Survey On Neural Word Embeddings*. DOI: 10.48550/arXiv.2110.01804. arXiv: 2110.01804 [cs]. URL: http://arxiv.org/abs/2110.01804. preprint.

Shao, Yang (Aug. 2017). "HCTI at SemEval-2017 Task 1: Use Convolutional Neural Network to Evaluate Semantic Textual Similarity". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. SemEval 2017. Ed. by Steven Bethard et al. Vancouver, Canada: Association for Computational Linguistics, pp. 130–133. DOI: 10.18653/v1/S17-2016.

Speer, Robyn, Joshua Chin, and Catherine Havasi (Feb. 12, 2017). "ConceptNet 5.5: An Open Multilingual Graph of General Knowledge". In: *Proceedings of the AAAI Confer-*

ence on Artificial Intelligence 31.1 (1). ISSN: 2374-3468. DOI: 10.1609/aaai.v31i1.11164.

Speer, Robyn, Joshua Chin, and Catherine Havasi (Dec. 11, 2018). *ConceptNet 5.5: An Open Multilingual Graph of General Knowledge.* DOI: 10.48550/arXiv.1612.03975. arXiv: 1612.03975 [cs]. URL: http://arxiv.org/abs/1612.03975. preprint.

Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum (May 8, 2007). "Yago: A Core of Semantic Knowledge". In: *Proceedings of the 16th International Conference on World Wide Web.* WWW '07. New York, NY, USA: Association for Computing Machinery, pp. 697–706. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242667.

Sánchez, David and Montserrat Batet (Mar. 1, 2013). "A Semantic Similarity Method Based on Information Content Exploiting Multiple Ontologies". In: *Expert Systems with Applications* 40.4, pp. 1393–1399. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2012.08.049.

Tai, Kai Sheng, Richard Socher, and Christopher D. Manning (May 30, 2015). *Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks.* DOI: 10.48550/arXiv.1503.00075. arXiv: 1503.00075 [cs]. URL: http://arxiv.org/abs/1503.00075. preprint.

Tian, Junfeng et al. (Aug. 2017). "ECNU at SemEval-2017 Task 1: Leverage Kernel-based Traditional NLP Features and Neural Networks to Build a Universal Model for Multilingual and Cross-lingual Semantic Textual Similarity". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017).* SemEval 2017. Ed. by Steven Bethard et al. Vancouver, Canada: Association for Computational Linguistics, pp. 191–197. DOI: 10.18653/v1/S17-2028.

Tien, Nguyen Huy et al. (Nov. 1, 2019a). "Sentence Modeling via Multiple Word Embeddings and Multi-Level Comparison for Semantic Textual Similarity". In: *Information Processing & Management* 56.6, p. 102090. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2019.102090.

– (Nov. 1, 2019b). "Sentence Modeling via Multiple Word Embeddings and Multi-Level Comparison for Semantic Textual Similarity". In: *Information Processing & Management* 56.6, p. 102090. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2019.102090.

Toshevska, Martina, Frosina Stojanovska, and Jovan Kalajdjieski (Apr. 25, 2020). "Comparative Analysis of Word Embeddings for Capturing Word Similarities". In: *6th International Conference on Natural Language Processing (NATP 2020)*, pp. 09–24. DOI: 10.5121/csit.2020.100402. arXiv: 2005.03812 [cs].

Turney, P. D. and P. Pantel (Feb. 27, 2010). "From Frequency to Meaning: Vector Space Models of Semantics". In: *Journal of Artificial Intelligence Research* 37, pp. 141–188. ISSN: 1076-9757. DOI: 10.1613/jair.2934.

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *Advances in Neural Information Processing Systems.* Vol. 30. Curran Associates, Inc.

Verma, Dhruv and S.N. Muralikrishna (July 2020). "Semantic Similarity between Short Paragraphs Using Deep Learning". In: *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT).* 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp. 1–5. DOI: 10.1109/CONECCT50063.2020.9198445.

Vrandečić, Denny and Markus Krötzsch (Sept. 23, 2014). "Wikidata: A Free Collaborative Knowledgebase". In: *Communications of the ACM* 57.10, pp. 78–85. ISSN: 0001-0782. DOI: 10.1145/2629489.

Wang, Liang et al. (Dec. 7, 2022). *Text Embeddings by Weakly-Supervised Contrastive Pretraining.* DOI: 10.48550/arXiv.2212.03533. arXiv: 2212.03533 [cs]. URL: http://arxiv.org/abs/2212.03533. preprint.

Wang, Zhiguo, Haitao Mi, and Abraham Ittycheriah (July 14, 2017). *Sentence Similarity Learning by Lexical Decomposition and Composition.* DOI: 10.48550/arXiv.1602.07019. arXiv: 1602.07019 [cs]. URL: http://arxiv.org/abs/1602.07019. preprint.

Wieting, John et al. (June 1, 2015). "From Paraphrase Database to Compositional Paraphrase Model and Back". In: *Transactions of the Association for Computational Linguistics* 3, pp. 345–358. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00143.

Wilkerson, John and Andreu Casas (May 1, 2017). *Large-Scale Computerized Text Analysis in Political Science: Opportunities and Challenges.* DOI: 10.1146/annurev-polisci-052615-025542. URL: https://papers.ssrn.com/abstract=2968080. preprint.

Wolf, Thomas et al. (Oct. 2020). "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.* Ed. by Qun Liu and David Schlangen. Online: Association for Computational Linguistics, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.

Young, Tom et al. (Nov. 24, 2018). *Recent Trends in Deep Learning Based Natural Language Processing.* DOI: 10.48550/arXiv.1708.02709. arXiv: 1708.02709 [cs]. URL: http://arxiv.org/abs/1708.02709. preprint.

Yu, Yong et al. (July 1, 2019). "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computation* 31.7, pp. 1235–1270. ISSN: 0899-7667. DOI: 10.1162/neco_a_01199.

Zad, Samira et al. (Oct. 2021). "A Survey of Deep Learning Methods on Semantic Similarity and Sentence Modeling". In: *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).* 2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 0466–0472. DOI: 10.1109/IEMCON53756.2021.9623078.

Zheng, Tao et al. (Aug. 7, 2019). "Detection of Medical Text Semantic Similarity Based on Convolutional Neural Network". In: *BMC Medical Informatics and Decision Making* 19.1, p. 156. ISSN: 1472-6947. DOI: 10.1186/s12911-019-0880-2.

Zhu, Ganggao and Carlos A. Iglesias (Jan. 2017). "Computing Semantic Similarity of Concepts in Knowledge Graphs". In: *IEEE Transactions on Knowledge and Data Engineering* 29.1, pp. 72–85. ISSN: 1558-2191. DOI: 10.1109/TKDE.2016.2610428.

Zhuang, Fuzhen et al. (Jan. 2021). "A Comprehensive Survey on Transfer Learning". In: *Proceedings of the IEEE* 109.1, pp. 43–76. ISSN: 1558-2256. DOI: 10.1109/JPROC.2020.3004555.

Zuccon, Guido et al. (Dec. 8, 2015). "Integrating and Evaluating Neural Word Embeddings in Information Retrieval". In: *Proceedings of the 20th Australasian Document Computing Symposium.* ADCS '15. New York, NY, USA: Association for Computing Machinery, pp. 1–8. ISBN: 978-1-4503-4040-3. DOI: 10.1145/2838931.2838936.

Řehůřek, Radim and Petr Sojka (May 22, 2010). "Software Framework for Topic Modelling with Large Corpora". In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks.* Valletta, Malta: ELRA, pp. 45–50.

A Appendix

A.1 Example Edges

Edge Content	Label
Kidnapped U.S. journalist missing in Syria for six weeks	-
Germany A Generation of Uncertainty	-
Rival Sudans hold summit on Friday, signal concessions	-
In El Salvador, Tooth Decay Blamed on Junk Food, Lack of Information	-
Greeks fight back against corruption	-
Tentative steps: Pakistani girl shot by Taliban leaves hospital to live with family in UK	-
Indian police to officially charge 5 suspects with murder in gang-rape case	-
Turkey accuses EU of bigotry, says its reforms are ignored	-
The Basque nationalist party Batasuna, outlawed in Spain in 2003, is closing its doors in France	-
Latvia is hailed as proof that austerity can work	-
UK Council for Child Internet Safety will draw up a list of approved sites offering advice about topics such as sex and drugs	-
North Korea praises knife attack on US ambassador to South Korea	-
Typhoon Goni lashes Japan	-
Bosnian Serb paramilitary leader gets 45 years for war crimes	-
We're Not Hacking You, Promise	-
Boko Haram violence affects millions of schoolchildren	-
China takes cautious step towards carbon emissions trading	-
Myanmar military plane crash kills four	-
North Korea condemns Japan's 'black-hearted' whaling industry	-
Major Political Donors Have Access to TPP Documents	-
South Korea Urges North to Be Serious Before Talks	-
Abandonment and Empty Streets Characterize Crisis in Rio	no conflict
Sister says don't make missing Flight 370 pilot the fall guy	no conflict
Taliban Announce Spring Offensive in Afghanistan	conflict
Israel warns Lebanon to curb Hezbollah reprisals for air strike	conflict
Bainimarama says NZ easing of sanctions 'insincere	no conflict
Iran's Nuclear Program May Have Cost the Country $500 Billion or More	no conflict
Paddy Power sponsors Dennis Rodman return trip to North Korea	no conflict
Avramopoulos is visiting Slovenia to discuss the migrant crisis	no conflict
Oh NO NOT Again: "Toyota Recalls 369K Vehicles Globally	no conflict
Air raids on Syrian city of Raqqa kill 39	no conflict

Table A.1: Randomly sampled exemplary edges contents of the Worldnews-SH, the Filtered Base Edge Set and the Conflict Dataset (in that order). Edges from the Conflict Dataset are listed with their corresponding labels.

A.2 Reference Edge Sets

Num. of Ref. Edges	Ref. Edges Set ID	Reference Edge Content
1	1-1	Israeli gunfire wounds Gaza fisherman: ministry
1	1-2	Ukraine's Opposition Accuses Government of Provoking Violence
1	1-3	Thursday's attack by armed youths on the base in Bor left at least 58 dead, including children
1	1-4	Turkey suspends 15,200 education staff
1	1-5	Kurdish protesters storm the Conservative Party's campaign headquarters in London
3	3-1	Israeli gunfire wounds Gaza fisherman: ministry Kuwait Rejects Saudi Request for War Subvention Researchers Accuse Canadian Internet Company of Helping Yemen Censor the Web
3	3-2	Ukraine's Opposition Accuses Government of Provoking Violence Chinese island-building in the South China Sea is causing "irreversible and widespread damage to biodiversity and ecological balance," according to the Philippines; Manila accused China of disregarding the people who rely on the sea by destroying coral reefs to create new islands The government's opposition and various refugee organizations have harshly criticized the reforms
3	3-3	Thursday's attack by armed youths on the base in Bor left at least 58 dead, including children Venezuela expels 3 US consular officials Russia, China nix U.S. human rights claims: Russia and China disputed U.S. Ambassador Nikki Haley's contention that human rights violations are a main driver of conflicts
3	3-4	Turkey suspends 15,200 education staff Leading Muslim groups condemn ISIS killing of US journalists PayPal freezes Canadian media company's account over story about Syrian family
3	3-5	Kurdish protesters storm the Conservative Party's campaign headquarters in London Obama seeks new Syria strategy review to deal with ISIS Tougher Canadian visa policy hits foreign workers, protects Canadian jobs

Table A.2: Edge content for the reference edge sets of size $N \in \{1, 3\}$

Num. of Ref. Edges	Ref. Edges Set ID	Reference Edge Content
10	10-1	Israeli gunfire wounds Gaza fisherman: ministry Kuwait Rejects Saudi Request for War Subvention Researchers Accuse Canadian Internet Company of Helping Yemen Censor the Web Afghanistan President Ashraf Ghani slams Pakistan for harbouring terrorists, praises India 5-year-old Kentucky boy fatally shoots 2-year-old sister Bangladesh police kill 'mastermind' of Dhaka cafe attack Philippines President Duterte orders army to destroy Islamic militants or risk ISIS disease Turkish jets kill 18 Daesh terrorists in northern Syria Report slams Israel's military law enforcement system Iran Pursuing Release of Sailors Abducted by Somalian Pirates
10	10-2	Ukraine's Opposition Accuses Government of Provoking Violence Chinese island-building in the South China Sea is causing "irreversible and widespread damage to biodiversity and ecological balance," according to the Philippines; Manila accused China of disregarding the people who rely on the sea by destroying coral reefs to create new islands The government's opposition and various refugee organizations have harshly criticized the reforms Syria and Russia oppose unilateral US strikes against ISIL in Syria Taliban attack in Afghanistan kills six policemen Turkish President condemns US commandos photographed sporting Kurdish militia insignia France could ease ban on gay men giving blood after ECJ ruling Libyan smuggler fighting kills 22 migrants Kazakhstan jails online editor for 'spreading false information Russia urges Assad to give up chemical weapons

Table A.3: Edge content for the reference edge sets of size $N = 10$ (Part 1/3)

Num. of Ref. Edges	Ref. Edges Set ID	Reference Edge Content
10	10-3	Thursday’s attack by armed youths on the base in Bor left at least 58 dead, including children Venezuela expels 3 US consular officials Russia, China nix U.S. human rights claims: Russia and China disputed U.S. Ambassador Nikki Haley’s contention that human rights violations are a main driver of conflicts Turkish PM tells female reporter to ’know your place South Korean prosecutors seek arrest of ex-President Park in corruption probe Taliban Announce Spring Offensive in Afghanistan Spain dismantles ’jihadist cell U.S. conducts ’counter terrorism strike’ against al Qaeda-linked target in Libya Swiss prosecutors launch money-laundering probe against fugitive Ukrainian President Yanukovych, and son UK Wants 10 Year Prison Sentence For Online Pirates
10	10-4	Turkey suspends 15,200 education staff Leading Muslim groups condemn ISIS killing of US journalists PayPal freezes Canadian media company’s account over story about Syrian family U.S. urges China’s Xi to extend non-militarization pledge to all of South China Sea Sri Lanka accuses Canada of holding Commonwealth ’to ransom Russia and pro-Moscow rebels on Wednesday condemned Ukraine for ratifying two bills on greater autonomy for the separatist east, saying they violated a peace deal and threatened a shaky month-long truce Malaysia turns away 800 boat people; Thailand spots 3rd boat US expresses concern over security of Pakistan’s Nuclear weapons Health experts accuse WHO of ‘egregious failure’ on Ebola Sunni militants accuse the army, perhaps the only widely respected public institution in Lebanon, of siding with Hezbollah

Table A.4: Edge content for the reference edge sets of size $N = 10$ (Part 2/3)

Num. of Ref. Edges	Ref. Edges Set ID	Reference Edge Content
10	10-5	Kurdish protesters storm the Conservative Party's campaign headquarters in London Obama seeks new Syria strategy review to deal with ISIS Tougher Canadian visa policy hits foreign workers, protects Canadian jobs U.S. preparing new sanctions against Chinese entities over financial support to North Korea Turkey's Erdogan makes Nazi jibe over Germany rally ban: "Your practices are not different from the Nazi practices of the past Brazil committee recommends Dilma Rousseff's impeachment U.S. dismisses Russian concern about missile defense system in South Korea South Korea mulls ban on bosses messaging employees at home Israel to destroy homes of Palestinian Jerusalem car attackers Majority of Finns reject NATO membership

Table A.5: Edge content for the reference edge sets of size $N = 10$ (Part 3/3)

A.3 Best F1-score based Eval. Run Comparison Tables

Evaluation Run Name				Prec.	Rec.	(Best) F1-Score	
CP	NM	RES	t_s				Std. Dev.
semsim-ctx	e5	r-1-1	0.65	0.386	0.768	0.514	-
semsim-ctx	e5	r-1-2	0.67	0.442	0.769	0.562	-
semsim-ctx	e5	r-1-3	0.59	0.372	0.838	0.515	-
semsim-ctx	e5	r-1-4	0.67	0.363	0.804	0.500	-
semsim-ctx	e5	r-1-5	0.68	0.397	0.681	0.502	-
semsim-ctx	e5	r-1-mean	0.64	0.356	0.823	0.477	+/- 0.046
semsim-ctx	e5	r-1-mean-best	0.65	0.392	0.772	0.518	+/- 0.025
semsim-ctx	e5	r-3-1	0.68	0.398	0.836	0.539	-
semsim-ctx	e5	r-3-2	0.68	0.435	0.752	0.551	-
semsim-ctx	e5	r-3-3	0.69	0.410	0.846	0.553	-
semsim-ctx	e5	r-3-4	0.69	0.392	0.846	0.536	-
semsim-ctx	e5	r-3-5	0.68	0.361	0.814	0.500	-
semsim-ctx	e5	r-3-mean	0.69	0.417	0.753	0.534	+/- 0.024
semsim-ctx	e5	r-3-mean-best	0.68	0.399	0.818	0.536	+/- 0.021
semsim-ctx	e5	r-10-1	0.71	0.415	0.817	0.550	-
semsim-ctx	e5	r-10-2	0.72	0.435	0.793	0.562	-
semsim-ctx	e5	r-10-3	0.72	0.400	0.771	0.527	-
semsim-ctx	e5	r-10-4	0.72	0.454	0.735	0.562	-
semsim-ctx	e5	r-10-5	0.71	0.378	0.835	0.520	-
semsim-ctx	e5	r-10-mean	0.72	0.428	0.746	0.543	+/- 0.021
semsim-ctx	e5	r-10-mean-best	0.72	0.416	0.790	0.544	+/- 0.020
semsim-ctx	gte	r-1-1	0.60	0.349	0.794	0.485	-
semsim-ctx	gte	r-1-2	0.63	0.388	0.799	0.522	-
semsim-ctx	gte	r-1-3	0.57	0.303	0.972	0.462	-
semsim-ctx	gte	r-1-4	0.55	0.300	1.000	0.461	-
semsim-ctx	gte	r-1-5	0.61	0.339	0.829	0.482	-
semsim-ctx	gte	r-1-mean	0.60	0.327	0.870	0.474	+/- 0.013
semsim-ctx	gte	r-1-mean-best	0.59	0.336	0.879	0.483	+/- 0.025
semsim-ctx	gte	r-3-1	0.64	0.364	0.826	0.505	-
semsim-ctx	gte	r-3-2	0.67	0.378	0.713	0.494	-
semsim-ctx	gte	r-3-3	0.67	0.385	0.740	0.506	-
semsim-ctx	gte	r-3-4	0.66	0.376	0.820	0.516	-
semsim-ctx	gte	r-3-5	0.62	0.322	0.896	0.474	-
semsim-ctx	gte	r-3-mean	0.66	0.374	0.714	0.486	+/- 0.040
semsim-ctx	gte	r-3-mean-best	0.65	0.365	0.799	0.499	+/- 0.016
semsim-ctx	gte	r-10-1	0.67	0.377	0.869	0.526	-
semsim-ctx	gte	r-10-2	0.67	0.361	0.917	0.518	-
semsim-ctx	gte	r-10-3	0.69	0.382	0.834	0.524	-
semsim-ctx	gte	r-10-4	0.69	0.399	0.730	0.516	-
semsim-ctx	gte	r-10-5	0.68	0.388	0.708	0.501	-
semsim-ctx	gte	r-10-mean	0.68	0.378	0.803	0.513	+/- 0.009
semsim-ctx	gte	r-10-mean-best	0.68	0.382	0.812	0.517	+/- 0.010

Table A.6: Best F1-score evaluation runs for all ERS of the form `semsim-ctx NM r-N-X` (CNESS with AT option disabled)

Evaluation Run Name				Prec.	Rec.	(Best) F1-Score	
CP	NM	RES	t_s				Std. Dev.
semsim-ctx	e5-at	r-1-1	0.68	0.338	0.915	0.494	-
semsim-ctx	e5-at	r-1-2	0.71	0.434	0.701	0.536	-
semsim-ctx	e5-at	r-1-3	0.66	0.356	0.885	0.508	-
semsim-ctx	e5-at	r-1-4	0.71	0.365	0.798	0.501	-
semsim-ctx	e5-at	r-1-5	0.71	0.351	0.908	0.507	-
semsim-ctx	e5-at	r-1-mean	0.69	0.350	0.847	0.487	+/- 0.021
semsim-ctx	e5-at	r-1-mean-best	0.69	0.369	0.841	0.509	+/- 0.016
semsim-ctx	e5-at	r-3-1	0.72	0.363	0.846	0.508	-
semsim-ctx	e5-at	r-3-2	0.72	0.389	0.765	0.516	-
semsim-ctx	e5-at	r-3-3	0.73	0.399	0.780	0.528	-
semsim-ctx	e5-at	r-3-4	0.73	0.386	0.829	0.526	-
semsim-ctx	e5-at	r-3-5	0.72	0.351	0.886	0.503	-
semsim-ctx	e5-at	r-3-mean	0.73	0.392	0.740	0.511	+/- 0.016
semsim-ctx	e5-at	r-3-mean-best	0.72	0.378	0.821	0.516	+/- 0.011
semsim-ctx	e5-at	r-10-1	0.74	0.382	0.849	0.527	-
semsim-ctx	e5-at	r-10-2	0.75	0.395	0.781	0.525	-
semsim-ctx	e5-at	r-10-3	0.75	0.405	0.815	0.541	-
semsim-ctx	e5-at	r-10-4	0.74	0.371	0.890	0.523	-
semsim-ctx	e5-at	r-10-5	0.74	0.357	0.881	0.509	-
semsim-ctx	e5-at	r-10-mean	0.75	0.395	0.766	0.521	+/- 0.016
semsim-ctx	e5-at	r-10-mean-best	0.74	0.382	0.843	0.525	+/- 0.012
semsim-ctx	gte-at	r-1-1	0.65	0.342	0.826	0.483	-
semsim-ctx	gte-at	r-1-2	0.68	0.386	0.774	0.515	-
semsim-ctx	gte-at	r-1-3	0.66	0.311	0.945	0.468	-
semsim-ctx	gte-at	r-1-4	0.63	0.300	1.000	0.461	-
semsim-ctx	gte-at	r-1-5	0.66	0.339	0.866	0.487	-
semsim-ctx	gte-at	r-1-mean	0.66	0.328	0.882	0.476	+/- 0.018
semsim-ctx	gte-at	r-1-mean-best	0.66	0.335	0.882	0.483	+/- 0.021
semsim-ctx	gte-at	r-3-1	0.69	0.340	0.827	0.482	-
semsim-ctx	gte-at	r-3-2	0.69	0.309	0.950	0.467	-
semsim-ctx	gte-at	r-3-3	0.72	0.348	0.852	0.494	-
semsim-ctx	gte-at	r-3-4	0.71	0.367	0.822	0.508	-
semsim-ctx	gte-at	r-3-5	0.67	0.317	0.930	0.473	-
semsim-ctx	gte-at	r-3-mean	0.69	0.324	0.883	0.472	+/- 0.013
semsim-ctx	gte-at	r-3-mean-best	0.70	0.336	0.876	0.485	+/- 0.017
semsim-ctx	gte-at	r-10-1	0.71	0.342	0.879	0.492	-
semsim-ctx	gte-at	r-10-2	0.72	0.336	0.910	0.491	-
semsim-ctx	gte-at	r-10-3	0.73	0.349	0.891	0.501	-
semsim-ctx	gte-at	r-10-4	0.72	0.341	0.885	0.492	-
semsim-ctx	gte-at	r-10-5	0.70	0.322	0.934	0.478	-
semsim-ctx	gte-at	r-10-mean	0.72	0.341	0.854	0.486	+/- 0.002
semsim-ctx	gte-at	r-10-mean-best	0.72	0.338	0.900	0.491	+/- 0.008

Table A.7: Best F1-score evaluation runs for all ERS of the form `semsim-ctx NM-at r-N-X` (CNESS with AT option enabled)

A.4 Evaluation Metric Scores vs. Similarity Threshold Plots

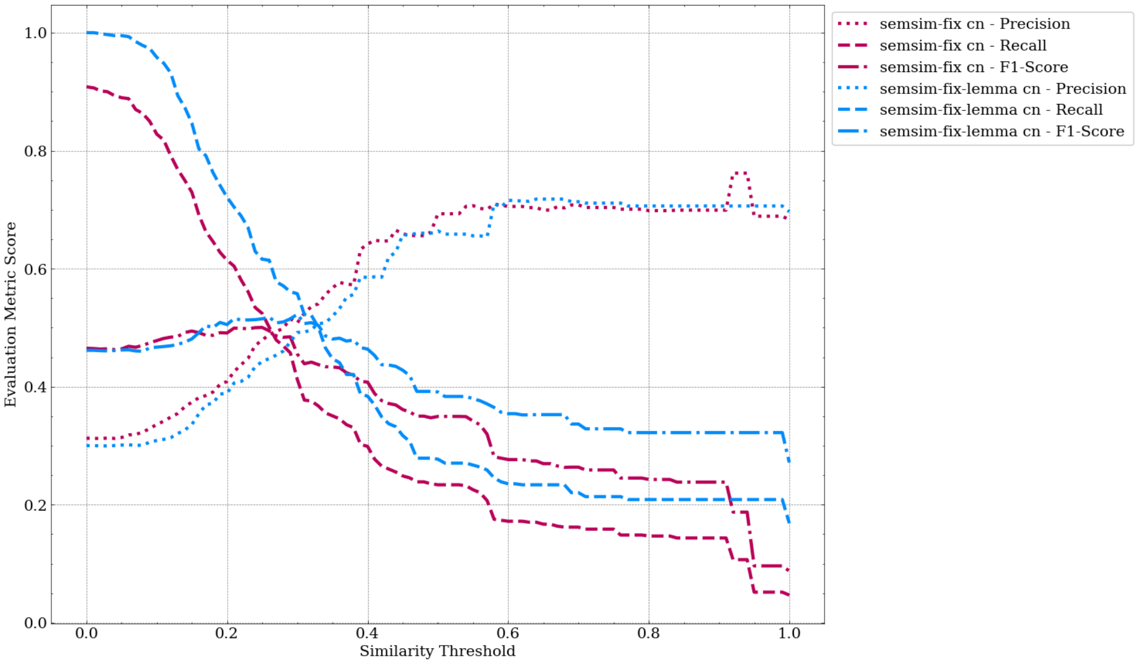


Figure A.1: Precision, recall and F1-score vs. ST for the evaluation runs `semsim-fix cn` and `semsim-fix-lemma cn`

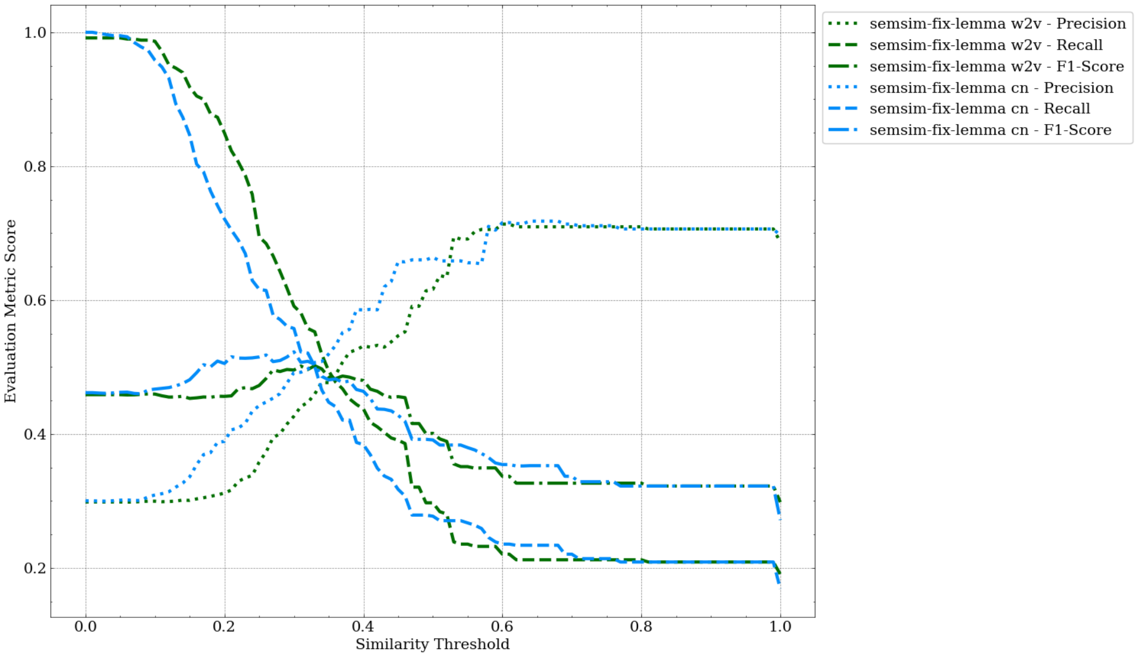


Figure A.2: Precision, recall and F1-score vs. ST for the evaluation runs `semsim-fix-lemma w2v` and `semsim-fix-lemma cn`

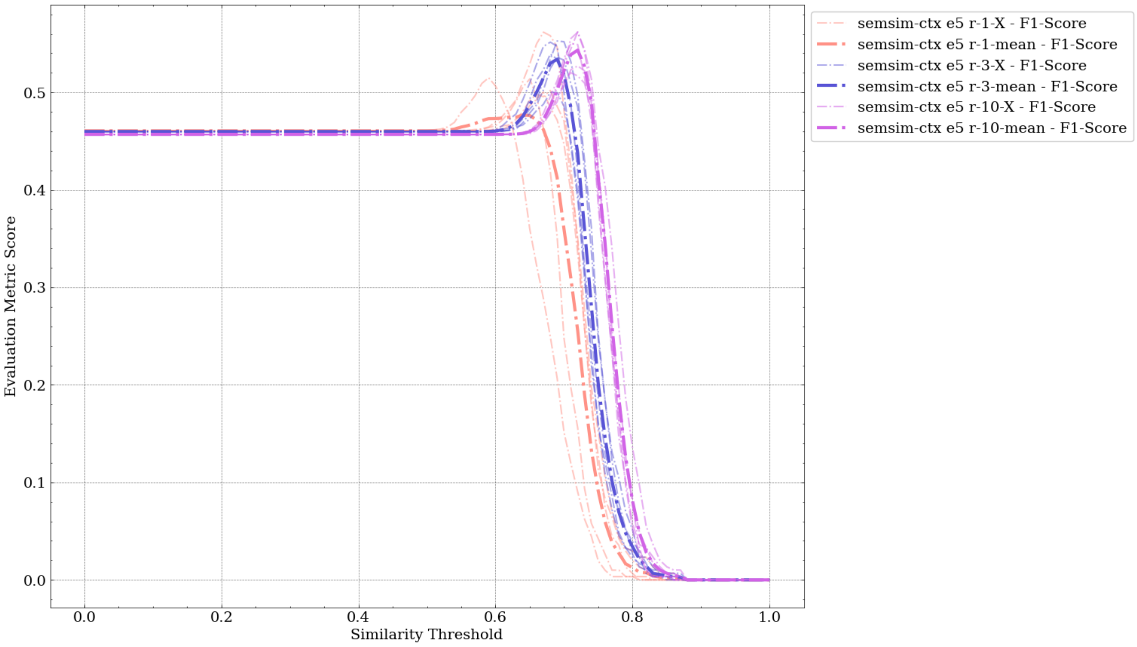


Figure A.3: F1-score vs. ST for the evaluation runs `semsim-ctx e5 r-1-X`, `semsim-ctx e5 r-3-X` and `semsim-ctx e5 r-10-X`

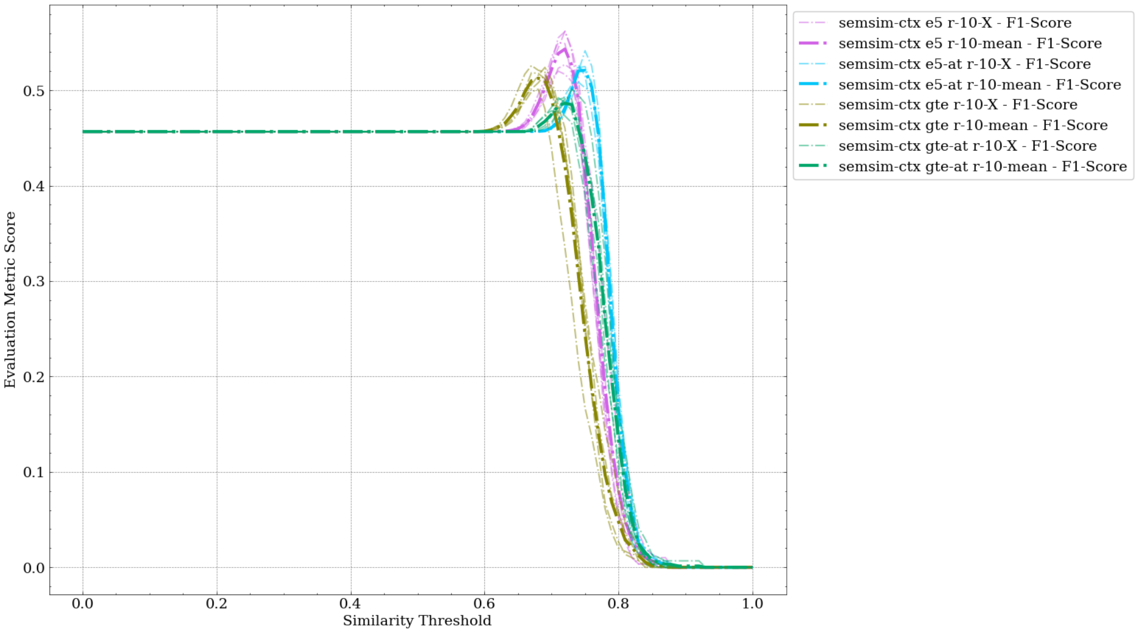


Figure A.4: F1-score vs. ST for the evaluation runs `semsim-ctx e5 r-10-X`, `semsim-ctx e5-at r-10-X`, `semsim-ctx gte r-10-X` and `semsim-ctx gte-at r-10-X`

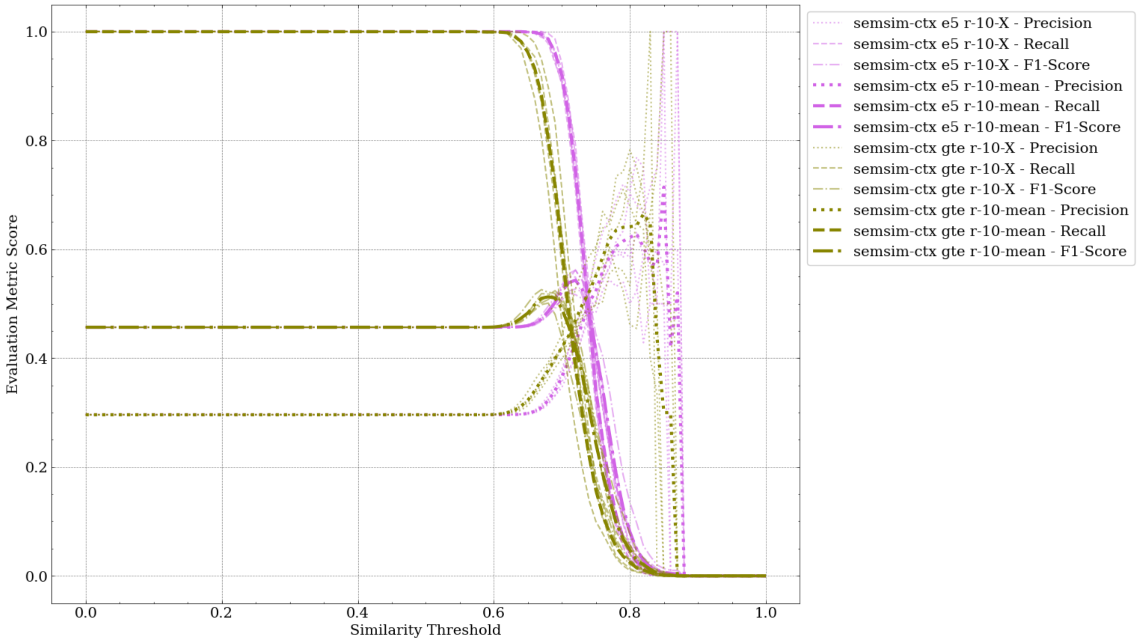


Figure A.5: Precision, recall and F1-score vs. ST for the evaluation runs `semsim-ctx e5 r-10-X` and `semsim-ctx gte r-10-X`

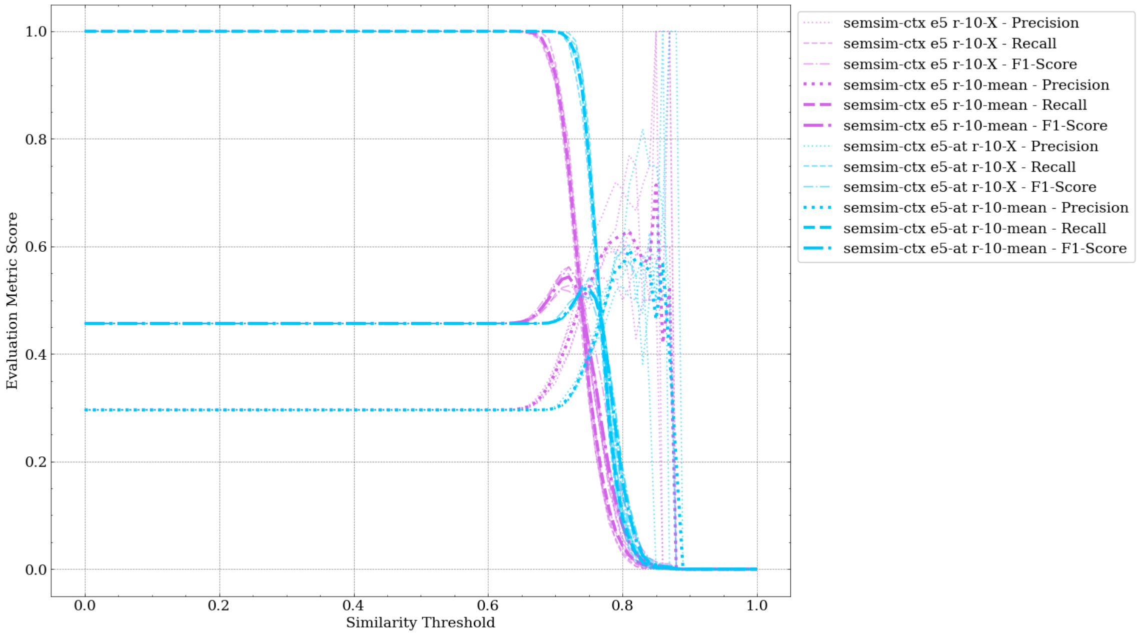


Figure A.6: Precision, recall and F1-score vs. ST for the evaluation runs `semsim-ctx e5 r-10-X` and `semsim-ctx e5-at r-10-X`

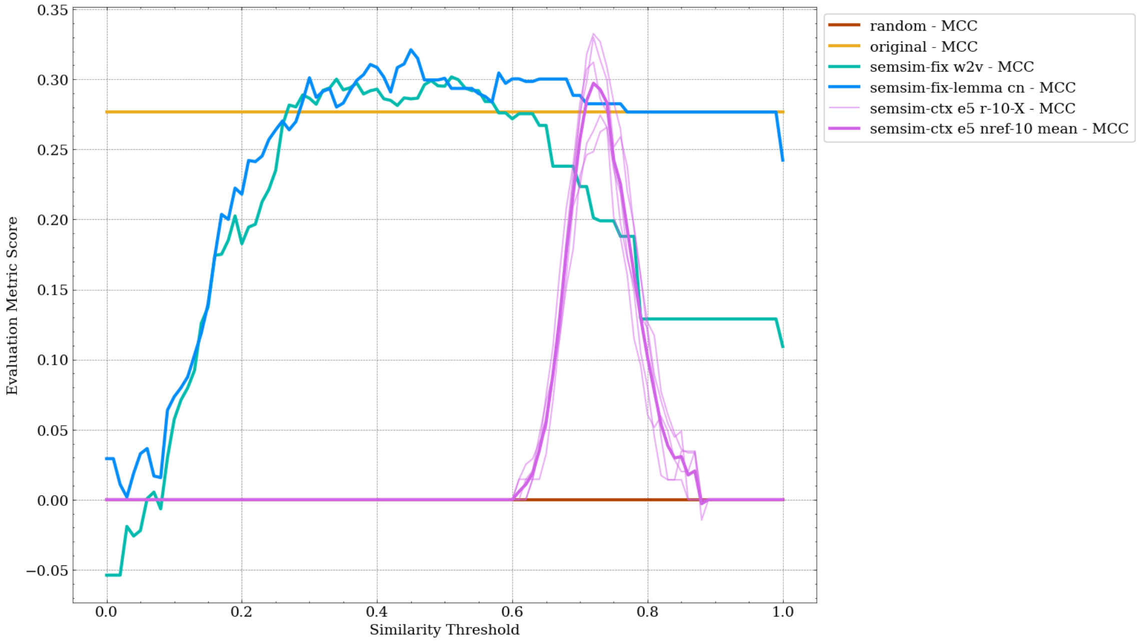


Figure A.7: Matthews Correlation Coefficient (MCC) vs. ST for the evaluation runs `random`, `original`, `semsim-fix w2v`, `semsim-fix-lemma cn` and `semsim-ctx e5 r-10-X`