UNIVERSITY OF COPENHAGEN
DEPARTMENT OF NORDIC STUDIES AND LINGUISTICS

Master Thesis

Nathalie Carmen Hau Sørensen

Can a Machine have Sense for Senses?

Building sense embeddings for the Danish Language

Supervisor: Bolette Sandford Pedersen

Submitted on: 31/05-2021

Re-edited on: 14/11-2021

Name of department:	Department of Nordic Studies and Linguistics (NORS)
Author(s):	Nathalie Carmen Hau Sørensen
Title and subtitle:	Can a Machine have Sense for Senses? Building sense embeddings for the Danish Language
Supervisor:	Bolette Sandford Pedersen

This is a edited version. I reworked some sections after finding minor typos and other mistakes.

Abstract

In this thesis, we investigate the optimal approach for creating Danish sense representations from a combination of word embedding models and handcrafted lexical resources. The aim of such a sense inventory is to use it for automatic sense tagging, which could enable other semantic tasks like sense clustering or sentiment analysis. Automatic sense tagging requires machine-friendly representations of word ambiguity, that is: a sense inventory that is not so fine-grained that the ambiguity cannot be resolved, yet fine-grained enough to capture essential meaning distinctions. Although dictionaries provide a representation of senses, their sense distinctions are often too fine-grained and inconsistently structured. Another approach is to deduce senses from a word embedding model – a statistical model that vectorizes the representation of a word from its surrounding words in running text. Our main research question is: to what extend can we represent senses in a meaningful way by combining the two approaches.

We test two word embedding models (word2vec and BERT) in combination with one of two lexical resources: the Danish wordnet DanNet or the Danish Dictionary. In the intrinsic evaluation, the quality of the sense representations is tested across a broad number of words, while the extrinsic evaluation tests the practical application of the sense inventory in a Word Sense Disambiguation task (WSD) of 18 highly ambiguous words. The results show that the word2vec and BERT models perform similarly and relatively poorly in the intrinsic evaluation. In the WSD task, the BERT model does achieve the best performance, which proves that the model is successful when given sufficient data. If we can provide sufficient training data for more senses, we expect the BERT model to be useful for sense-tagging for more words. The findings highlight why it is important to collect sufficient data for a lower-resource language, like Danish, if we want to achieve high quality results.

Acknowledgement

*I would like to give my **deepest thanks** to my supervisor **Bolette Sandford Pedersen** for their guidance, both in my free topic and in this thesis.*

*I would also like to thank my study comrade **Conrad** who has been on the same journey as me.* **Thank you for all the moral support and discussions**.

And of course, thanks to all my kind friends and family members who took the time to proofread this thesis in the final stressful moments.

*Finally, I would like to express my gratitude to all the **dedicated people** that works towards **improving Danish language technology**. Especially thanks to the **Centre of Language Technology**[1] and the **Society for Danish Language and Literature**[2]. This thesis would not have been possible without your important work in Danish lexical semantics.*

[1] https://cst.ku.dk/english/

[2] https://dsl.dk/

Table of Contents

1 Introduction

What is the meaning of the word *face*? A simple question on the surface, but the answer is not straightforward. A word can have many meanings, also called senses. If presented alone, a word with multiple senses is ambiguous. However, the senses of ambiguous words are usually distinguishable by using the context a word is used in, as we can see in the examples below:

(1a) *She has a long, thin **face**. (The front of the head)*

(1b) *He thinks he would lose **face** if he admitted the mistake. (Honour, respect)*

(1c) *She abseiled down the rock **face**. (The front of an object)*

Humans can intuitively separate the different senses of *face* in the examples above, probably without being actively aware of the many other possible meanings. Since technology plays an increasing role in our lives, computers need to learn how to understand language. That includes the complex relationship between how words are used, and what they mean. The question then is, how do we teach computers to capture the different senses of a word? Is it possible to efficiently and automatically sense-tag natural language? With this work, we hope to improve our answer to those questions regarding the Danish language.

One possible first step could be to create representations of word senses that we can feed into the machine. Humans commonly represent meanings in lexical resources like dictionaries, in which meanings are defined through entries. Whilst dictionaries do contain valuable information about a word and its senses, it can be challenging using machines to model that information. First, dictionaries tend to have fine-grained sense distinctions that can be difficult for computational systems to capture (Pedersen et al., 2018). Additionally, dictionaries are handcrafted by multiple lexicographers with a human reader in mind, which makes them prone to inconsistencies in the sense distinctions and descriptions[3] (Kilgarriff,1997; Kilgarriff, 2007).

[3] An entry in a corpus-based dictionary is based first and foremost on corpus evidence, but it also uses the intuition that the lexicographer has from their individual mental lexicon. The mental lexicons of two lexicographers are not completely identical since they have different experiences of language.

Nowadays, dictionaries are created on a foundation of language evidence by using a corpus-based approach. We can do something similar by automatically modelling language from a large corpus using word embedding models. These models represent the linguistic input as vectors in a dense vector space (Jurafsky & Martin, 2018; Goldberg, 2016). The placement of a word in the vector space reflects its distributional position in the language since the word vector is computed from the distribution of words in the surrounding context. Therefore, the word vector (or word embedding) is a representation of all the contexts, that word appears in. This might not be enough to capture all the information that is present in a lexical resource[4]. Additionally, some word embedding models (static) fail to create separate representations for words with multiple senses. One solution is to represent senses as a combination of the distributional information from the word embedding model and the knowledge from a lexical resource. In this way, we create dense vector representations of word senses that can effectively distinguish the multiple senses of ambiguous words. In this work, we plan to examine how different types of word embedding models (static and contextualised) can be combined with different knowledge from lexical resources. Besides being useful for sense-tagging, the sense representation can also help detecting which senses from the lexical source that could be suitable for a future clustering[5].

When working with meaning, it is difficult to determine what a meaning is precisely. It is a problem that has been up for discussion among philosophers, linguists, and other researchers for many years (Grice, 1969; Firth, 1957; Wittgenstein, 1953; Kilgarriff, 1997). Not disregarding the importance of the discussion, our problem is more akin to the practical challenge of compiling machine-readable representations of meaning. Additionally, we aim at making representations that computers can process in a similar way that humans process meaning. That may not entail that a ma-

[4] Word embedding models do not explicit model different sense relations and may be lacking real world knowledge.

[5] This is one of the goals in semantic part of the COR project: https://cst.ku.dk/english/projects/the-central-word-register-for-danish-cor/

chine will understand language and meaning as humans do. However, by mimicking human behaviour, a system can be used as a tool for analysing natural language – both for researchers to better understand semantic patterns in language data and helping the public with good language technology.

This is particularly important when working with a lower-resource language like Danish. Natural languages can differ in how concepts are expressed. For example, languages split the colour spectrum up in different categories (Cook, Kay, & Regier, 2005) and have different lemmas for the same concept[6]. From a linguistic perspective, a computational model can provide additional insight into how lexical semantics differs between languages. This can give us a better understanding of how meaning works in general. However, it also requires research in other languages than the higher resourced languages like English. Since the availability of relevant data differs between higher and lower resource languages, it is important to investigate which methods work with the available data. From a practical point of view, a language user should not be restricted on their access to high-quality language technology based on their native language. For instance, a Danish speaker should not have to switch to English to use technology like Google Translate or Siri, because of the research gap between the Danish and English language technology. We hope this work will be one step closer to narrowing the research gap in language technology between a lower resourced language (Danish) and the higher-resourced languages.

1.1 Focus

The thesis focusses on examining the optimal approach for creating Danish word sense representations using a combination of word embedding models and lexical resources. We do so by experimenting with two types of word embedding models: the static and contextualized models. Due to time constraints, we are not investigating every available word embedding model, but limit ourselves to one model for each embedding model type; word2vec (Mikolov et al., 2013) as static model and BERT (Devlin et al., 2018) as contextualised model. The purpose is to probe into the

[6] A typical example is the many words for snow in Inukitut popularized by Franz Boas

potential in using deep learning methods, in particular the contextualised BERT model, in Danish computational lexical semantic research.

The arguments behind examining both static and contextualised embedding models are (a) their different level of complexity and (b) their different procedure in computing representations. Static word embedding models like the word2vec (Mikolov et al., 2013) was developed first and have a simple architecture and are faster to train with less data. They create a single static representation of a word based on all the different contexts the word appears in. Thus, the model combines all the meanings of a word into the same representation. However, this leaves them ill-equipped for representing semantic phenomena like homonymy and polysemy that causes disparities between the meaning and form in words (Engberg-Pedersen et al., 2019). However, we could work around the problem by using the static embedding model as foundation for creating separate representations for each word sense. We divide these senses by first embedding sense-specific information from a lexical resource, and then calculate the average embedding of the sense information. This is what we do in the thesis. We combine the information extracted from the Danish wordnet DanNet (Fellbaum 1998, Pedersen et al., 2009) and the word2vec model (Sørensen & Nimb, 2018; Mikolov et al., 2013) to make separate representations for each sense in DanNet.

In contrast with static models, contextualised embedding models create dynamic representations from the context a word appears in. If a word sense is clearly distinguishable from the context, this model will capture such different senses of the word. However, that also requires a more complex architecture that usually takes longer time to train and more data. The BERT model (Devlin et al., 2018) is a contextualised model which achieved state-of-the-art results on a wide range of tasks. Because of the results of the BERT model, it is interesting to explore the potential of the model on Danish lexical semantics. Such a study on Danish has not been carried out to our knowledge, and therefore we wish to take the first steps in examining the sense representations generated by a BERT model pretrained on Danish[7]. We experiment

[7] https://github.com/certainlyio/nordic_bert

with both sense representations utilizing example sentences only and a combination of example sentences and other information from DanNet.

The aim is to enlighten the most promising direction for future work with lexical semantics and word embedding models. Therefore, we evaluate the sense representations both intrinsically and extrinsically. The intrinsic evaluation uses the Danish Word-in-Context dataset (DanWiC) (Hau Sørensen, 2021), which tests whether the sense representations can distinguish senses of the same lemma given two contexts. This can inform us how the sense representations of the same lemma relate to each other. The extrinsic evaluation is done by a Word Sense Disambiguation (WSD) task using a subset of the SemDaX dataset, a Danish semantic dataset annotated with senses from the Danish Dictionary (Pedersen, et al., 2016). The WSD task tests the application of the sense representations in a practical setting, to see if they can be used for appropriate Danish sense tagging. The finding of the thesis is intended as help for ongoing projects and future projects within Danish language technology.

In the evaluation, we limit the types of systems in which we integrate the sense representations. We use (a) a cosine similarity-based system that directly compares the representations in the semantic vector space, and (b) a trained neural network classifier. The cosine similarity-based system does not require additional training after the sense representations have been generated and directly measures the quality of the semantic vector space. A neural network classifier requires labelled training data, but it can tell us whether the sense representations carry any learnable patterns of information.

1.2 Problem statement and research questions

This thesis investigates how Danish sense representations can be generated by combining information from a statistical approach (word embedding models) and handcrafted lexical resources (DanNet, The Danish Dictionary). We examine combinations of lexical resources with two types of word embedding models: static and contextualised.

We aim to answer the following research questions:

Q1 To what extent can we create meaningful sense representations by combining the embedding models with handcrafted lexical resources?

Q2 In combination with a lexical resource, which of the two embedding models (type-based or contextual) work the best for sense-tagging?

The different approaches to generating sense representations are evaluated intrinsically using DanWiC, a Danish Word-in-Context task, and extrinsically using a Word Sense Disambiguation task (WSD) performed on a subset of SemDaX.

1.2.1 Hypotheses

In the intrinsic evaluation, we expect the sense representations using a contextualised embedding model to be better at distinguishing senses of the same lemma than the static representations, since (a) these models are naturally larger and more advanced, and (b) they do not conflate multiple senses into the same representation.
However, we also expect the fusion of DanNet information with the static embeddings to help separating the senses from each other. We hope both all the different types of sense representations can enlighten which kind of information from DanNet is the most useful when distinguishing senses.

In the extrinsic evaluation, we expect a pipeline with a contextualised model to be the best at sense-tagging in the WSD task. This is based on the fact that the combination of DanNet and static embeddings (word2vec) have shown poor performance on a WSD task in earlier experiments (Olsen et al, 2020). Meanwhile, in recent WSD experiments, a contextualised model (Yap et al, 2020, Bevilacqua & Navigli, 2020) has reached the new state-of-the-art on English.

1.3 Contributions

The work of the thesis contributes the following:

- Knowledge of the implications of different implementations of sense representation systems for Danish.
- Experimental results from computational models on the DanWiC task that can be used to indirectly evaluate lexical resources.
- An implementation of a system for WSD for Danish using BERT and DanNet (or another lexical resource).
- Laying the foundation for future work with word embedding models in Danish computational semantics.

1.4 Thesis outline

The thesis consists of seven chapters, which includes the current introductory chapter.

- In chapter 2, we introduce the theoretical background of computational lexical semantics. This includes a presentation of WSD, knowledge resources, and embedding models.
- Chapter 3 describes the material used for the experiments.
- Chapter 4 presents the methodology of creating sense representations from word embedding models and lexical resources.
- Chapter 5 goes through the intrinsic and extrinsic evaluation.
- Chapter 6 presents and analyse the results.
- Chapter 7 is a general discussion of the results from the previous chapter and problems with the approaches.
- We finish off with a conclusion and suggestions for future work in chapter 8.

1.5 Terminology

BERT	Bidirectional Encoder Representations from Transformers
CoS	Cosine Similarity Approach
FT	Fine tuned
Lemma	We use *lemma* to talk about the abstract form of a word. This is similar to the term *word type*. Since we have a focus on the lexical aspect of the word ambiguity problem, we chose to go along with *lemma*.
NN	Neural Network
NLP	Natural Language Processing
Sense representation	We use *sense representation*, *sense embedding* and *sense vector* interchangeably. They all refer to the vectors in the semantic vector space created by our models.
Token	*Token* or *word token* refers to a specific instance of a *lemma* in a corpus
WSD	Word Sense Disambiguation

2 Theoretical Background

The theoretical framework of the thesis is a partnership between the semantic subfield of linguistics and statistical, machine learning (ML) approaches from computer science. Therefore, this work lies within Natural Language Processing (NLP): the interdisciplinary field between computer science and linguistics that studies the interaction between computers and human language. The goal is to make computational systems understand human language through the processing of natural language data.

In general, there have been two dominant paradigms in NLP: the knowledge-based methods and data-driven, statistical methods (Jurafsky & Martin, 2018). The knowledge-based methods take inspiration from formal linguistics where language is seen as a productive system (e.g., Chomsky's generative grammar (1957)) Their logical formalisation of language is motivated by the resemblance to the logic of computers and the simplicity and efficiency that this brings. These approaches rely on the manual encoding of linguistic knowledge for language processing. In contrast, data-driven, statistical methods use corpora and statistical modelling to acquire linguistic knowledge. With the rise of ML and neural networks, the data-driven approach has become dominant in NLP. This work is an attempt to combine both methods to create sense representations from lexical resources (the knowledge base) and word embedding models (learnt language representations).

In this chapter, we will start by introducing computational semantics. Then, we will begin in the linguistic camp, by giving an account of word senses and lexical resources. Hereafter, we will explain the distributional approach to modelling word senses, which includes a description of word embedding models.

2.1 Computational semantics

Computational semantics is the study of meaning through computational methods. This is an important subfield of NLP, since understanding language is central for tasks like information retrieval, machine translation, and sentiment analysis. Meaning in natural language can be expressed at multiple levels (e.g., words, phrases, sentences, and discourse). In this work, we focus on lexical semantics, the study of

meaning on the word level. We aim to build word sense representations that can be used for distinguishing and tagging senses.

Sense tagging is the process of assigning a particular sense from a sense inventory to every content word in a text (Wilks & Stevenson, 1997). Sense tagging attempts to mark every token in a text[8]. A token is an individual instance of a word. This is distinct from the word type, or lemma, which is that same word abstracted from any particular instance. For instance, the English tokens *walk*, *walks*, *walked* belongs to the same lemma: *walk*. Since, we are working with lexical resources, we use lemma over word type[9].

2.2.1 Word sense ambiguity

A sense is a distinct concept that a lexeme maps onto (Cruse, 2011). Multiple concepts (i.e., meanings) can be expressed by the same lemma, which causes a one-to-many mapping between lemmas and senses. Therefore, a key challenge for sense tagging is to disambiguate sense.

In lexical semantics, the multiple senses of an ambiguous lemma constitute the phenomena of homonymy and polysemy (Engberg-Pedersen, Boye, & Harder, 2019). Homonymy occurs when two lemmas share a common form, whilst their senses are unrelated. This usually results in different lexical entries in dictionaries (Cruse, 2011, p. 115). Polysemy is when the multiple senses of a lemma are related. There are two categories of polysemy: linear and non-linear. Two senses are linearly related if one sense is a restriction on the semantic area of the other sense, for instance a subclass (hyponym) or a part (meronym)[10]. If there is no specification-generalisation relation between the two senses, they are said to be non-linearly related (Cruse, 2011, p. 115). Figurative senses are cases of non-linear polysemy where the figurative use is a metaphoric (by resemblance) or metonymic (by association) extension of the literal sense.

[8] How to handle function words is up to the system, for instance with as *O* in an IOB markup.

[9] The lemma comes from the tradition of lexicography, and therefore also refers to the word form in lexical entries.

[10] Hyponym and meronym are cases of semantic relations. See more under section 2.2.2.

One can analyse senses to different degrees of granularity through linear polysemy. One task for a lexicographer is to decide when a case is a subsense of another, and they may split or merge the same set of language use into different senses. A key decision for any semantic system is there to decide which sense granularity to use (Ide & Wilks, 2007). In NLP, several studies have shown that coarser sense distinctions result in higher quality systems, A machine does not necessarily have to detect the finer subsenses to get a basic understanding of the language, but just the senses that can cause confusion.

2.1.2 Word Sense Disambiguation for sense tagging

Word Sense Disambiguation (WSD) is the procedure of selecting the sense of a lemma that is activated by a context (Navigli, 2009). WSD is a key task in computational lexical semantics because of its relevance to the conflation of multiple senses into the same lemma, which causes word ambiguity. WSD has been classified as an AI-complete problem (Mallery, 1988; Navigli, 2009). This assumes that a strong artificial intelligence (AI) on the level of the human mind is necessary to solve the task.

WSD is related to sense tagging, as the disambiguation of senses is a prerequisite for tagging ambiguous lemmas. We distinguish WSD from sense tagging, as a WSD is a broader class of algorithms that can be used as a tool for sense tagging (Wilks & Stevenson, 1997). Additionally, WSD tasks have several variations. For instance, the goal of a lexical sample WSD algorithm is to disambiguate one target token in a context sentence from a limited set of target lemmas. In contrast, an all-words WSD aims at tagging every content word token in a sequence (Navigli, 2009). Further variations on WSD depend on how the senses are defined (e.g., sense inventory), the granularity of the sense inventory, and whether the WSD system is developed for a generic or specific domain. In this work, we work with a lexical sample WSD task with sense inventories from the lexical resources.

2.1.3 The problem of word senses for WSD

When we build a word sense representation system with the purpose of sense tagging and disambiguation, we need to determine what kind of sense inventory to use. The straightforward solution is to base our senses on lexical resources like dictionaries.

This prompts the discussion of whether we can define a general-purpose sense inventory from lexical resources (Kilgarriff, 1997).

The question is emphasized by taking a usage-based approach to meaning, inspired both by Grice (Grice, 1969). A word sense can be defined as a commonly accepted meaning (Navigli, 2009). For a meaning *m* of lemma *l* to be commonly accepted, a pairing between *m* and *l* must be present in the mental lexicon of a large enough proportion of the speakers in the given language community. The meaning of a word in a speaker's mental lexicon is an abstraction of their experience of it. Their experience comprises every utterance in which they have heard, read, or produced the word. Different patterns of use cause the process of abstraction (lexicalisation) of different meanings of the word in the mental lexicon.

This results in three problems, when working with a predefined sense inventory:

1. New senses can arise when a lemma is used in a new pattern. This dynamic expressivity of a lemma cannot be captured by a fixed sense inventory.
2. The boundary between lexical knowledge and real-world knowledge is unclear, as is the boundary between individual senses (e.g., when is a usage pattern different enough to entail a new sense). These are a problems if we consider our senses as discrete entities.
3. The mental lexicons of two or more speakers differ since their experiences differ. If the sense inventory is handcrafted by humans, there will be inconsistencies between how they split and define senses.

These problems are not limited to deciding on a sense inventory for a computational system. They touch on fundamental questions about how to represent senses in general. For our purpose of comparing approaches for building Danish sense representations, it is an advantage to have a fixed and finite sense inventory. First, this simplifies the comparison of systems. Secondly, our method will line up with a group of other computational semantic project in Danish that are all based on the same lexical resources (Pedersen et al. 2009; Pedersen, et al., 2016; Pedersen et al. 2018a).

2.2 Knowledge-based description of senses

The knowledge-based methods in NLP derive from formal linguistics due to their compatibility with the nature of computers. In traditional formal semantics, the strict focus on formal logic results in an emphasis on function words: the natural languages' version of logical elements. The interests were in truth-conditions, reference,

and causality. The content words in the lexicon were seen as an uninteresting appendage to the formal grammar. However, the need for the formalisation of the meaning of content led to the development of formal lexical theories and knowledge-based systems that attempted to structure the meaning of content words, like the AQUILEX-project (Copestake, 1990) and SIMPLE (Lenci, et al., 2000).

A prominent theoretical framework is the Generative Lexicon proposed by Pustejovsky (Pustejovsky, 1998; Pustejovsky, 1995). Instead of treating the lexicon as an enumerative listing of senses, like it is done in formal semantics (e.g., Montague semantics (Montague, 1970)), the lexicon is built on generative mechanisms that create different interpretations of lemmas based on regularities in the semantic structure. The mechanism that represents the core meaning of words is called Qualia Structure. The aim of the Qualia Structure is to represent different dimensions of meaning (Qualia roles), which can dissolve polysemy (Pustejovsky & Jezek, 2016). The dimensions of meaning capture the different properties of objects that are reflected in language use. The dimensions are shown in table 1.

Dimension	Description	Relations
Formal	Information about the basic conceptual category (i.e., taxonomic information) *What is it? What is its nature?*	Hyponymy, hypernymy, is-a, way-of
Constitutive	Information about parts/constituents *What is it made of? What are its constituents?*	Meronymy, holonomy, concerns, involves
Telic	Purpose and function information *it for? How does it function?*	Used-for, agent, patient
Agentive	Information about the origin *How did it come into being? What brought it about?*	Made-by, created-by

Table 1: The four dimensions of Qualia Structure (Pustejovsky, 1995).

The Qualia structure allows the decomposition of meaning, akin to compositional analysis. In compositional analysis, meaning is seen as a complex structure of a restricted set of primitive units (i.e., components or features). However, this method

cannot account for the expressiveness of language with a finite set of features. In the Generative Lexicon the decomposition of meaning focusses on how different terms relate to and influence each other's meanings. For instance, *car* may in different contexts evoke the relations *engine* (constitutive: has-part), *vehicle* (formal: is-a) and *wheels* (constitutive: has-part). A value for a dimension can be unspecified or have multiple possible values, which gives qualia structure its high level of generality.

2.2.2 WordNet as a sense inventory

A wordnet is a large computational lexicon structured as a network of concepts. The first wordnet, Princeton WordNet (Fellbaum, 1998) was created from psycholinguistic principles and was an attempt to portray the mental lexicon. It has since become a valuable resource of lexical information in language technology. The English WordNet has become the standard sense inventory for English WSD and annotated semantic datasets (Navigli, 2009).

A wordnet relies on relational semantics in which a word sense (and the corresponding concept) is determined by the set of its semantic relations to other senses. These relations include synonymy (same/similar concept), antonymy (opposite concept), meronymy (part-of), and hyponymy (subordination) (Cruse, 1986). The lexicon can therefore be structured as a network of senses that are interconnected by semantic relations (Quillian, 1967).

In a wordnet, a concept, or sense, is represented as a set of synonyms called a synset. A synset also contains a gloss, which contains a definition of a sense and sometimes a set of example sentences. In this way, the synsets contain similar lexical knowledge of senses as seen in a dictionary, though a wordnet does not have a structure of main and subsenses. The structure of the network is also a representation of

Formal Role	**Agentive Role (ORIGIN)**	**Constitutive Role (COMPOSITION)**	**Telic Role (PURPOSE)**
has_hyperonym	made_by (from SIMPLE)	has_holo_made_of	used_for (from SIMPLE)
has_hyponym		has_holo_part	used_for_object
is_a way_of		has_holo_member	role_agent
		has_holo_location	role_patient
		has_mero_made_of	
		has_mero_part	
		has_mero_member	
		has_mero_location	
		concerns (from SIMPLE)	
		involved_agent	
		involved_patient	
		involved_instrument	

Table 2: The DanNet relations on the four Qualia roles. From Pedersen et al. (2009, p. 287)

senses in a formal language as the semantic relations follow the Qualia structure. Table 2 shows the semantic relations of the Danish wordnet DanNet framed on Qualia structure. Knowledge about the ontological type is also added in some wordnets. DanNet follows EuroWordNet and uses their multidimensional Top Ontology (Vossen, 1998) that is based on Lyons' three-category structure (Lyons, 1979). Both the Qualia Structure and the ontology is a source of real-world knowledge.

2.3 Vector Semantics

Vector semantics is a technique to represent senses as vectors (i.e., a list of numbers) in a semantic space. The technique is also known as vector space modelling (VSM), and it has become the standard way of representing word meaning in NLP (Jurasky & Martin, 2018). VSM approaches use statistics to create the semantic space from word distributions in large text corpora. The word embedding model is a widely used vector representation since the introduction of the word2vec model (Mikolov et al., 2013). This section introduces the theory that motivated the work in vector semantics and word embedding models.

2.3.1 The distributional hypothesis

The origin of VSM is found in the *distributional hypothesis* which was formulated during the 1950s by linguists like Harris (1954) and Firth (1957). The core idea in the distributional hypothesis is that meaning similarity correlates with distributional similarity. Words with similar meaning tend to occur in similar contexts and therefore have similar distributions. By studying how words appear and relate to each other in large text corpora, we can gather information about their meanings. If the meaning of words can be found in their usages, then we should be able to automatically generate representations of meaning from the word distributions in large text collections. This is the foundation for building vector space models.

2.3.2 Vector Space models

A vector space model (VSM) uses linear algebra to represent linguistic units (e.g., a word sense) in a geometric space, also called a vector space (Turney & Pantel, 2010). The vector space is a multidimensional space that consists of a set of vectors V, a set of scalars (real numbers) also called a field $\mathbb{F}$, and two operations: *vector addition* and *scalar multiplication*. Each vector corresponds to a point in the space with a nu-

merical value at each dimension. Representing linguistic units in a vector space is a method for capturing linguistic information while also being able to compare elements with mathematical operations. This is most visible with the similarity between two units. The closer two vectors are in the vector space, the more similar the representations are, and the more similar we can consider the units according to the modelled information. We can measure the similarity between the representation with metrics like the *cosine similarity measure*. We can consider these models a type of mapping from the conceptual space of natural language to a mathematical space, where the linguistic units are expressed as vectors.

The vectors in a VSM can be organised into a matrix to handle the large number of representations generated from text corpora (Turney & Pantel, 2010). Following the distributional hypothesis, we can structure a VSM as a word-context matrix (Deerwester et al., 1990), where the lemmas correspond to the row vectors, and the contexts (e.g., a sentence) correspond to the column vector. Both the lemmas and the contexts can be represented in the same vector space. To disambiguate a sense in context, we can use a similarity measure to find the closest sense to a context.

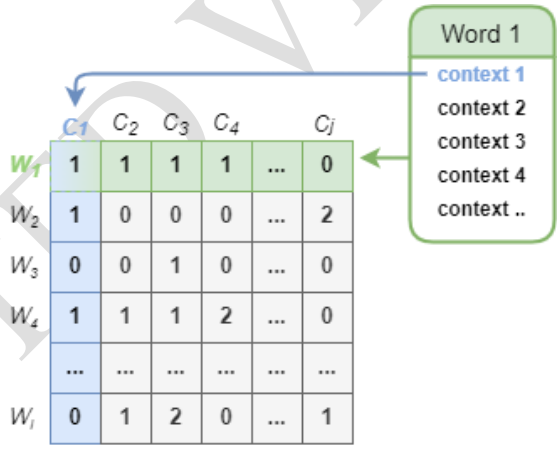


Figure 1: Word-context matrix (right). The blue column vector is a context representation, and the green row vector is a lemma representation.

The approach follows the *bag of words hypothesis* that estimates the meaning content of a context as the bag-of-words of that context. Mathematically, a bag (or multiset) is a set that allow duplicates. A bag-of-words representation only considers the unique types and their frequencies in the bag, thus any information expressed by word order is disregarded. In the word-context matrix, each element is a representation of the frequency of a given lemma in each document. Thus, we are essentially collecting information on how often a word occurs in different contexts, and then

transforming the information into a representation for every lemma. A word-context matrix can be seen in figure 1.

A disadvantage of a traditional VSM approach is the high dimensionality of the vectors. Since the number of dimensions corresponds to the length of the vocabulary and number of contexts, the dimensionality can easily reach millions. Since most of the dimensions include a high number of zeros, the representation will be sparse. The high number of dimensions and the sparse vectors cause the *curse of dimensionality* that generally cause inefficient computations and difficulty using the representations in machine learning. In recent years, it has shown to be useful to directly learn dense low-dimensional word representations through neural networks. These models are known as word embedding models, and they have become the new standard of NLP[11].

2.3.4 Short introduction to the neural network

A neural network is a powerful supervised learning model inspired by the human brain. Essentially, a neural network model is a mathematical function that maps a set of input values to output values (Goodfellow et al., 2016). This function is comprised of several simpler functions (e.g., matrix multiplication and activation function). Numerous architectures have been developed for neural networks. However, an example of a simple feed-forward neural network is sufficient to understand the core idea behind them. Figure 2 shows a fully connected feed-forward neural network. A neural network consists of layers of neurons (i.e., nodes): an input layer, one or more hidden layers, and an output. each layer is a new, further processed representation of the input data. The neurons in each layer have weighted connection to neurons in the next layer, illustrated by the lines on the figure.

[11] Both dimensionality reduction and word embedding models remove the interpretability of the dimensions in the word vectors.

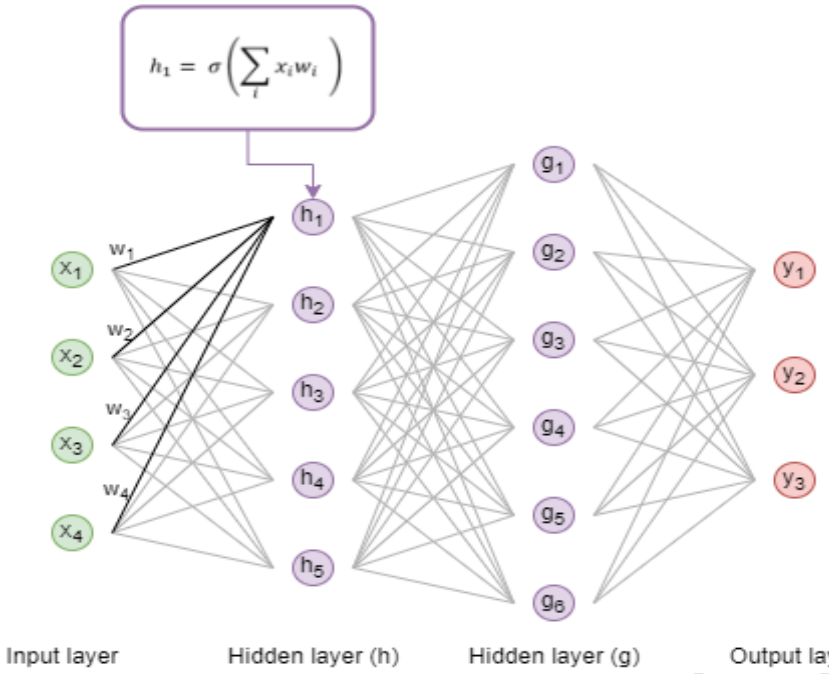


Figure 2: Fully connected feed-forward neural network with 2 hidden layers (*h* and *g*). σ is the activation function. x_i and w_i are respectively the input value and weight for neuron *i*. The network maps the 4-dimension input vector x to the 3-dimension output vector y.

Every neuron also contains an activation function, typically a non-linear function like the sigmoid[12]. The non-linear activation functions allow the neural network to learn non-linear patterns from the input data. If the input to the network moves in a single direction from the input towards the output layer, the network is called a feed-forward neural network. The weights of the model are determined during the training phase. An optimisation algorithm is used to explore the set of possible weights that a model can have to make good predictions. For this, we need a loss function that evaluates how well the model with the current weights performs on the data by returning the cost (or loss) of the weight setting. We can optimize the weights in the model by using the loss function and back-propagation (Rumelhart, Hinton, & Williams, 1986). In backpropagation, the gradient of the loss function is calculated, and the loss is distributed across the layers of the network. A backward pass can then be performed to adjust the weights of the model before we pass another training instance after which the process is repeated.

[12] The sigmoid function maps the input to a value between 0 and 1.

2.4.2 Static embedding models

An embedding is a representation of a feature vector in a dense, low dimensional vector space (Goldberg, 2016). Each context in the VSM word-context matrix can be considered a feature of the words, and thus translating the VSM into a dense vector space will create a word embedding model.

The first generation of word embeddings models are static models. These models learn a fixed representation of each lemma (word type) independently from the context a word may appear. The models are trained on a variety of different contexts. However, if a lemma occurs in widely different contexts, for instance by having multiple senses, the models will conflate the different contexts into a single representation. The static models are just look-up tables between a lemma and a representation.

2.4.2.1 Word2vec

The word2vec (Mikolov, 2013) is a widely used static embedding model (Camacho-Collados & Pilehvar, 2018). It uses a similar architecture to a neural network with some crucial differences. First, the aim, when training the word2vec model, is not to use the actual output of the network. Instead, the intention is to learn the weights of the hidden layer and use them as our representation. Secondly, the hidden layer only contains linear activation. That means that the word2vec representation is a projection of the input layer. The resulting hidden layer weight matrix has the shape $V \times D$, where V is the size of the vocabulary of the training data and D is the number of neurons in the hidden layer. We can compare the weight matrix with the VSM *word-context* matrix from earlier (see section 2.3.2). Each row in the matrix corresponds to a word in the vocabulary like in the VSM. However, instead of having possible contexts as columns, the contexts are represented in a continuous space of D dimensions, resulting in a dense representation.

The input layer in the word2vec model is a one-hot vector representation of the input word. When we calculate the value for each neuron in the hidden layer, which is a weighted sum of the input layer, only the weights for the index of the input word are returned, since all other values will be zero. Therefore, the hidden weight matrix is essentially a lookup table after we have set the weights. This can be seen in figure 3.

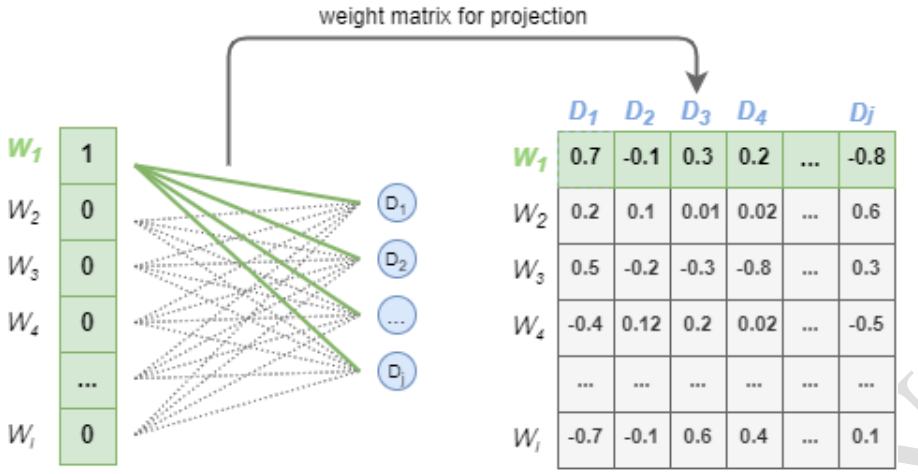


Figure 3: Projection weight matrix. The input word (green) is represented as a one-hot vector. When retrieving the representation from the weight matrix, the one-hot vector functions an index for the word representation (green row in matrix). The values in the grey rows are the weights of other words in the vocabulary. These are all zeroed out in the final output.

There are two different word2vec model architectures: Continuous Bag-Of-Words (CBOW) and skip-gram. They are trained on different, yet related tasks. An illustration of the tasks can be seen on figure 4.

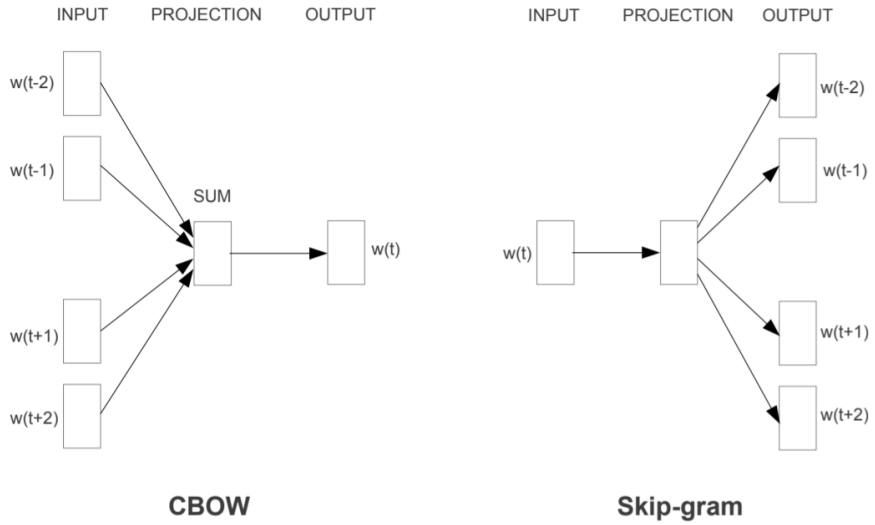


Figure 4: The two architectures of the word2vec model; CBOW and Skip-gram. CBOW is trained to predict a word from a context window, while Skip-gram predicts the surrounding context from a word. Figure from Mikolov et al. (2013, p. 5)

The CBOW's training objective is to predict a word given a set of words in a context window. This architecture is called a bag-of-words because the order of the words does not influence the projection (hidden layer). The input to the CBOW model is a combination of the one-hot vectors of the words in the context of a target word. The output is a probability distribution over the entire vocabulary. The highest value, that

is the dimension with the highest proportion of the probability distribution, is the predicted word for the context. The objective of the skip-gram model is the opposite of the CBOW. Instead of predicting a word given a context, it aims at predicting the words within a context window of given a word. In the skip-gram model, the context words are weighted by their proximity to the input word (Mikolov et al., 2013).

Both for CBOW and skip-gram, the window size is a super-parameter[13] that needs to be set. A window size of 5 is recommended for CBOW, while skip-gram benefits from a larger window size in the region of 10. Additional super-parameters include the dimensionality of the hidden layer (projection), which is typically between 100 and 1000, and the sub-sampling size[14].

2.4.3 Contextualised embedding models

A contextualised embedding model computes a dynamic representation of a word given a context. These models are based on the context a lemma appears in and not the lemma itself as in the typed-based models. Since the word representation changes with the context, the contextualised embedding models can avoid the problem of meaning conflation deficiency. Numerous contextualised embeddings have been proposed. The list includes: context2vec (Melamud, Dagan, & Goldberger, 2016) (Peters, et al., 2018) and BERT (Devlin et al., 2018). Due to constraints on time, available models for Danish, and computational power, we will focus on the BERT model.

2.4.3.1 BERT

The BERT model uses transfer learning, where the network is first pre-trained on a task, before the pre-trained network can be fine-tuned to another task, thereby basing the new fine-tuned model on the representations learnt from the first task (Devlin et al., 2018). The pre-training of BERT involves two unsupervised tasks: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). In MLM, 15% of the

[13] Parameters that determine the setup of the model.

[14] As highly frequent words (typically function words) carry little information, these can be sub-sampled to improve performance.

words in each sequence are masked by a [MASK] token. The aim of the model is then to predict the masked word based on the context of the unmasked words in the sequence. This task does not require any labels as the true value is already present in the training data. In NSP, two sentences are inputted to the model. Here, the aim is to predict whether the second sentence is the actual subsequent sentence in the original document. The model is trained on both tasks simultaneously by using their combined loss.

The name BERT is an abbreviation of its actual name *Bidirectional Encoder Representations from Transformers*. The terms Encoder and Transformers come from the network architecture seen in the deep layers of BERT. A Transformer is a type of sequence-to-sequence model using two related networks: an encoder and a decoder. The encoder captures the relevant information from all the tokens in the input sequence and represents it in the last hidden layer. The decoder then decodes the same last hidden layer. For instance, the encoder can create a representation of a sentence in language A, and the decoder can then translate the sentence into language B using the hidden representation from the encoder. Since the purpose of BERT is to generate a language representation, it only uses the encoder network.

The encoder Transformer network uses a method called self-attention. An attention mechanism computes relevance weights for every vector in a sequence in relation to a target vector. In self-attention, the sequence of vectors is the input sequence, and the target vector is one of the tokens from the input sequence. self-attention relates every position in the input sequence with every token in the same input sequence. A representation of a token is therefore a representation of itself and its weighted dependencies on the other tokens in the input. this finds which tokens from the same input sequence are the most relevant for interpreting the target token. Figure 5 is an illustration of how attention works.

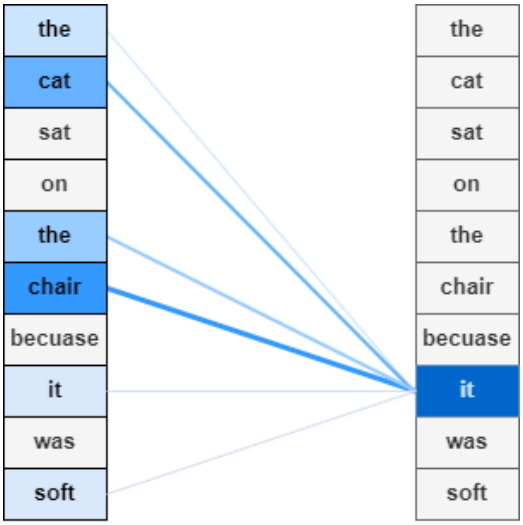


Figure 5: Illustration of how attention might work for the token *it*.

The attention used in transformers is called Scaled Dot-Product attention (Vaswani, et al., 2017). An illustration of the steps is shown in figure 6 to the left. The attention is calculated by first linearly projecting each token, which have already been embedded, into three vectors: the query, key, and value vectors. The query-key-value concept originates from information retrieval, where the query is the information needed, the key is a mapping of the information needed to the value, and the value is the information that we will retrieve. To select the relevant tokens, the query is then compared to every key from the input sequence to calculate the attention weights: the weight of how relevant a given key is to the target query. The attention weights are scaled and normalised to a probability distribution. This distribution determines how much the tokens are activated relative to each other. The final representation is calculated by weighting the value vector according to the activation distribution.

The attention is not computed for the input sequence in a single attention function. Instead, the attention is split between several attention heads that calculate the attention in parallel. The idea behind splitting the attention into several heads is to allow the model to jointly attend to information from representations in different subspaces and positions. This will allow the final representation to capture a wide range of information types from the input. This is called multi-head attention and it is the building block of the encoder transformer.

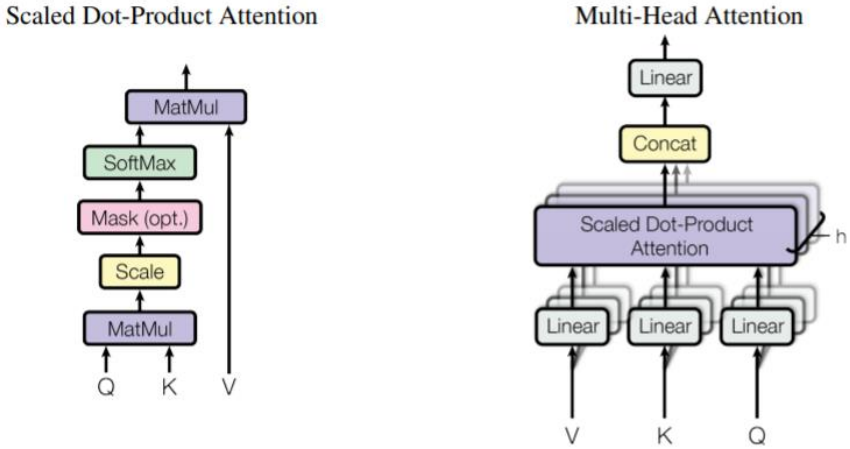


Figure 6: Scaled Dot-Product Attention (left) and Multi-Head Attention which consists of several layers running in parallel (right). Taken from (Vaswani, et al., 2017, p. 4)

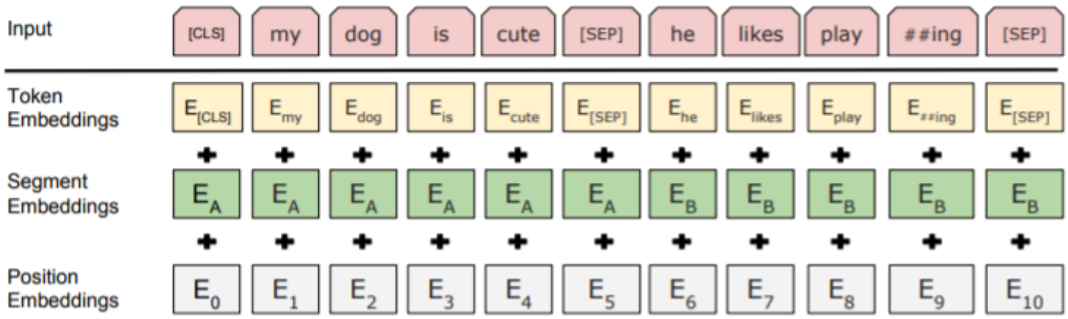


Figure 7: BERT input representation. Retrieved from Devlin et al. (2018, p. 5)

The BERT final architecture consists of an embedding layer, which is essentially a look-up layer between the tokens and a learned embedding, followed by several stacked Encoder Transformer layers. The input to the embedding layer is an input representation constructed from the sum of token embeddings, segment embeddings, and position embeddings as seen in figure 7. The token embeddings are a numerical representation of each token (words, subwords, or a special token).

The segment embeddings inform the model whether a token belongs to the first or second sentence, since BERT can represent two sentences at once. The positional embeddings are simply the position of a token in a sentence.

2.5 Sense representations for WSD

WSD relies on sources of linguistic knowledge to identify and represent senses. This knowledge typically comes from corpora (both annotated and unannotated) or lexical resources like machine readable dictionaries and wordnets.

The approaches using corpora are part of the statistical paradigm of NLP. These approaches can also be seen as a method for Word Sense Induction (Camacho-Collados & Pilehvar, 2018), the process of uncovering the senses that are present in the corpora data. One example is Schütze's (1998) context-group discrimination that represents senses from clustering of contexts with ambiguous lemmas. A context is calculated as the average (centroid) of the words it contains, all represented as word vectors. The context vectors are then clustered into a predetermined number of context groups (clusters). The Multiple-Sense Skip-Gram (MSSG) (Neelakantan et al., 2015) builds on the idea of representing the senses of a target lemma from the clusters of context vectors. By extending the word2vec Skip-gram model, the MSSG approach learns both the clustering (sense representation) and the word embeddings used in the context vector.

The approaches using lexical resources belong to the knowledge-based paradigm. They utilize the content and structure from lexical resources to infer the senses from the context. An advantage of this paradigm is the larger coverage of senses that they adopt from the lexical resources. However, knowledge-based approaches typically perform worse than the distributional corpus-based ones (Navigli, 2009).An example of a knowledge-based approach is the Lesk algorithm (Lesk, 1986), which disambiguates lemmas by selecting the sense that has the most overlap with the surrounding words of the target lemma. The original algorithm uses dictionary definitions; however, it has also been extended by Banerjee and Pedersen (2003) to explicitly add semantic relations from a wordnet to the sense definitions.

A third hybrid paradigm has also emerged. These approaches typically train a word embedding model on an unannotated corpus. Sense embedding are then obtained by using a lexical resource to split the word embedding space into a semantic space. Johansson and Nieto Pina (2015) do this by using a modified word2vec Skip-gram model, where sense information is added from the Swedish semantic network SALDO (Borin, Forsberg, & Lönngren, 2013) during training. Olsen et al. (2020) also use a word2vec model, however their process of building sense representations is separate from the training of the embedding model. Like the extended Lesk model (Banerjee and Pedersen, 2003), they represent senses with information from a wordnet (DanNet) which is embedded using word2vec. The final sense representation is then a centroid of the embedded information from DanNet. There is also a group of models that use BERT and wordnet glosses (Yap, Jie, & Chng, 2020). They fine-tune the pre-trained BERT on an annotated corpus, though with information from WordNet (Fellbaum, 1998). They create context-gloss pairs from the corpus and WordNet to create binary training instances on which the train the model on. Thus, the model has direct knowledge from the lexical resource.

3 Materials

The materials of the thesis are presented in the following sections.

3.1 Lexical resources

The lexical resources lay the foundation for investigating word senses in Danish. Though it is impossible to create a finite set of word senses for a language (see section 2.1.3), they do provide an estimation of the most common senses in Danish for many lemmas. In this work, we use two lexical resources: the Danish Dictionary (Den Danske Ordbog, DDO) and the Danish wordnet, DanNet (Pedersen et al., 2009). DanNet was used as the primary resource, however we deem it necessary to include additional information from DDO in the extrinsic evaluation because of the restricted definition (character limit of 25) and lack of fixed phrases in DanNet.

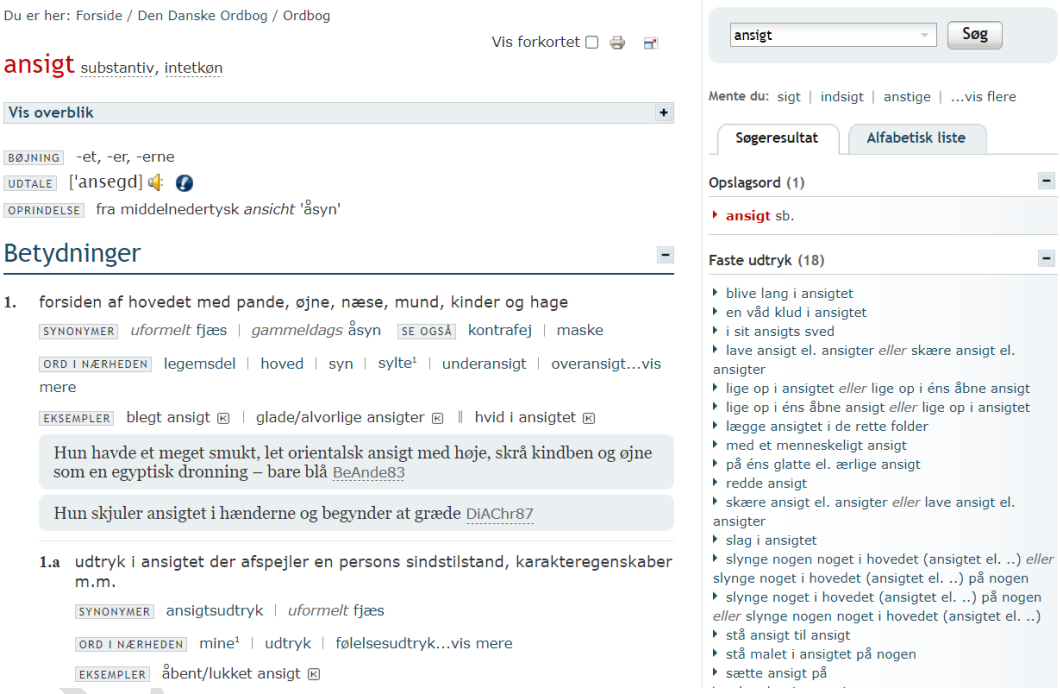


Figure 8: Screen shot of an entry for the lemma 'ansigt' (face) in the Danish Dictionary (DDO).

3.1.1 The Danish Dictionary (DDO)

The Danish Dictionary (DDO) is a corpus-based dictionary created by the Society of Danish Language and Literature, DSL[15]. The 2020 online version contains over

[15] The website of the Society of Danish Language and Literature: https://dsl.dk/

130,000 senses described in approximately 100,000 entries. Additional 11,000 fixed phrases are also listed. the dictionary also includes over 42,000 example sentences (i.e., contexts) for the senses.

The dictionary is based on Danish language from the 1950s to present time. The largest contribution comes from the electronic corpus that was specially created for the construction of the dictionary. Therefore, the dictionary and the example sentences are heavily influenced by corpus examples from the 10-year period 1983-1992. The sources contain text from among others: news articles, magazines, books, television, spoken-language interviews, parliamentary debates, commercials, and letters. The online version of the dictionary is continuously updated. In this work, the dictionary is used to get example sentences or definitions where the wordnet was not sufficient.

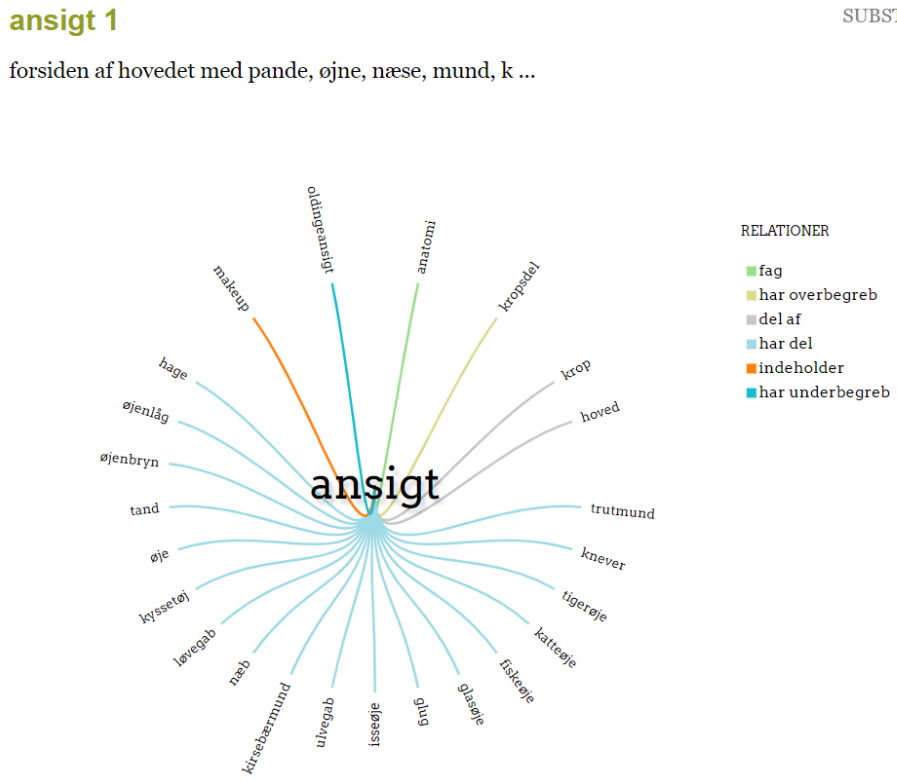


Figure 9: Visualisation of one of the synsets related to the lemma 'ansigt' (face) in DanNet.

3.1.2 DanNet – the Danish WordNet

DanNet is a wordnet for the Danish language that was semi-automatic compiled from the Danish Dictionary (DDO) (Pedersen et al., 2009). As the Princeton WordNet (Fellbaum, 1998), DanNet is structured around synsets: a set of synomyms denoting a concept inter-related to other concepts in the network by semantic relations.

Currently, DanNet contains 66,308 concepts represented as synsets with 326,566 links from semantic relations. In this work, we extract the lemmas, relations (represented as the lemma(s) of the synset), the shortened definition, and example sentences.

3.2 Datasets

3.2.1 DanWiC – Word-in-Context dataset for Danish

DanWiC is a Danish word-in-context dataset compiled from DanNet (Hau Sørensen, 2021). The aim of a word-in-context dataset is to test the ability of a model to differentiate the senses related to the same lemma. The problem is framed as a binary classification task, where the input consists of a target word and two contexts, and the output is whether the target word was used with the same sense in the two contexts. The contexts are the example sentences found in DanNet. If the two contexts come from the same synset, the label of the instance is 1 for positive, and if not, the label is 0 for negative.

The dataset is split into four subsets: training, development, test, and survey. In the development and test set, the labels are balanced, no lemma occurs more than three times, and every context sentence is unique. The survey is a collection of instances from the test set that has been evaluated by 8 human participants.

3.2.2 SemDaX – Semantic Danish dataset

SemDax is a collection of human-annotated data for Danish (Pedersen, et al., 2016). The data is extracted from the CLARIN Reference Corpus with data from different text types, including news articles, magazines, discussions, parliament debates, blogs and chat forums. A subcorpus of the semDaX data contains sense annotations with senses from DDO on 18 highly ambiguous nouns. These nouns are a collection of the nouns in the DDO with the highest number of main and sub-senses while also being highly frequent.

An instance in this subset of semdax contains 1) a target word, 2) a context sentence with the target word, and 3) one or more sense labels. Each sentence is annotated by 2-6 annotators, who are either advanced students or researchers. In the cases where the annotators disagree, multiple senses are present in the sense labels, hence why a sentence can have more than one labelled sense.

In total, 5569 annotated sentences with the target nouns are used in this project. Table 3 lists the 18 target nouns used in this project.

Target noun	Number of senses	Number of sentences
Ansigt (face)	16	401
Blik (look, glance, tin)	8	208
Hold (team, side, gang)	10	250
Hul (hole, gap, leek)	22	276
Kontakt (contact, switch, touch)	9	177
Kort (card, map, plan)	21	249
Lys (light, candle, lamp)	30	536
Model (model, pattern, design)	9	151
Plade (plate, sheet, disc)	13	367
Plads (room, space, square)	21	366
Selskab (company, party, society)	11	127
Skade (harm, injury, magpie)	12	290
Skud (bullet, guess, injection)	20	397
Slag (battle, stroke, cape, roll)	28	375
Stand (state, shape, booth, stand)	11	315
Stykke (piece, part, length)	22	533
Top (top, peak, apex)	12	260
Vold (violence, control, bailey)	10	290

Table 3: The target nouns and their number of senses in the dataset and sentences. The number of sentences is calculated as $100 + 15 * \text{number of senses}$, or so close to that with the available data (Pedersen, et al., 2016).

3.3 Models

3.3.1 Static model – Danish word2vec

We use the Danish word2vec model that is created by the Society for Danish Language and Literature (Sørensen & Nimb, 2018), The model is trained using the Gensim[16] implementation of Skip-gram with a window size of five. The training dataset consist of approximately 920 million running words from newswire (primary source), magazines, transcripts from the Danish Parliament, and fiction. All text originates from the time span between 1982 to 2017. In total, the corpus includes 6.3 million types (lemmas), with roughly five million types having a frequency below five.

3.3.2 BERT

The Danish BERT model is created by the company BotXO[17] as part of the Nordic BERT project. The training of the model is done using 8 v3 TPUs with a batch size of 1280, max sentence length of 256 tokens, 1 million training steps, and a learning rate of 2e-5.

The Danish training corpus is compiled from different online sources. These in a common crawl (9.5 Gb), The Danish Wikipedia (221 Mb), the two biggest Danish debate forums (dindebat.dk, 123 Mb; hestenettet.dk, 45 mb), and lastly Danish OpenSubtitles (881 mb). In total, the corpus consists of approximately 1.6 billion tokens[18] and 19.5 million unique tokens.

[16] https://pypi.org/project/gensim/

[17] https://github.com/botxo/nordic_bert

[18] We use tokens as the corpus also included special characters, though only ASCII characters besides the Danish letters æ, ø, å.

4 Methodology

The following chapter will present how word embedding models and lexical resources are combined into sense representations. We introduce two types of sense representations:

1. Danish word2vec combined with information from DanNet synsets.
2. Pretrained BERT[14] that we fine-tune, and that uses one or more example sentences from a lexical resource to model a sense representation.

The first type (word2vec) explicitly models the knowledge from DanNet by computing the centroid word2vec embedding of all the tokens in extracted information. To differentiate between the sense representation types, we call this the *static representation*, because it uses a static embedding model (word2vec).

The second type (BERT) only uses example sentences or definitions. This limits what kind of knowledge from DanNet, we can use in the embedding model. However, we also assume that the embedding model has already learnt to separate senses during the pretraining and fine-tuning, which will make the extra knowledge from DanNet redundant. We consider this an implicit modelling of senses and will call it the *contextualised representation*.

Before presenting how the static and contextualised representations are created, we first mention some practical challenges that have influenced the methodology. Then, we present the problem of fixed phrases and how it is handled. We also describe key pre-processing steps. Thereafter, we will explain the approaches to making the static and contextualised sense representations. Lastly, we will explain the evaluation method and the application of sense representations.

4.1 Practical challenges

Major limiting factors in most machine learning approaches include the available data and the computational power. Both factors play a crucial role in this thesis. First, working with Danish is a challenge as it is a lower-resource language, which means that the available open-source data is limited compared to a higher-resource language

like English. Additionally, with the constraints of time and access to GPUs, it was not possible to pretrain the BERT model from scratch, and therefore we use the open-source Danish pretrained BERT model[19]. The optimal comparison of embedding models requires that the models be trained on the same data, which is not the case (see section 3.3 for a description of the training data). As an alternative, we could have trained a word2vec model on the same data as the BERT model, but this data was not completely publicly accessible. In the end, the best option considering the constraints is using pretrained models with the consequence of a possible confounding variable from different training data.

The evaluation method is also heavily influenced by the availability of data. All the example sentences in the DanWiC dataset are taken from DanNet, which means that we are using the same data to generate the sense representations and to test the quality of those representations. To our knowledge, there are no other intrinsic evaluation datasets for Danish that allow us to evaluate contextualized embedding models. Therefore, we continue using the DanWiC dataset with this possible bias in mind.

The same problem does not occur in SemDaX as all the sentences come from the Clarin Corpus, however they use DDO as sense inventory. To test the static representations that are based on DanNet, we need a mapping to DanNet. There is no official open-source key between the resources. Another mapping does exist for the SemDaX senses. However, this mapping has a major problem by not handling fixed phrases.

4.2.1 Handling fixed phrases

A fixed phrase consists of a string of words that together form a sense that cannot be predicted from the sum of the words. An English example of a fixed phrase is *to kick the bucket* that means 'to die'. This sense cannot be inferred from any of the senses of *kick* or *bucket.* The question is if we should consider the fixed phrases established enough in the language to form their own lemma-meaning relation or see it as a frequent context for a lemma with the meaning being one of the possible senses of that lemma. In the framework of sense disambiguation, these fixed expressions are com-

[19] https://github.com/botxo/nordic_bert

mon in the language and can easily be distinguish by their fixed forms. DanNet does not contain any synsets that represent fixed phrases[20]. Therefore, a mapping to a DanNet synset reduces the form of the fixed phrases to the lemma with the literal sense, thus making the disambiguation task harder by removing the advantage of the fixed phrases: the restricted form. In the SemDaX WSD task, we only map to DanNet senses with sense representation that requires a DanNet sense inventory (experiment 4, experiment 5; word2vec). Though, we also test another framework that treat the fixed phrases as their own senses (experiment 5; BERT).

4.3 Preprocessing

All sentences from DanNet, DanWiC, and SemDaX are preprocessed. The preprocessing steps vary depending on which embedding model is used. Common for both models is the removal of special characters. This is done because the sentences from the lexical resources contain special notation related to the lexical entries. For instance, tokens are added to some example sentences to make them comprehensible outside their respective texts. These changes are marked with brackets. We also aim for consistency between the input for the two embedding models. Some typical abbreviations in the DanNet definition and example sentences are also expanded (e.g., *el.* → *eller*). The disadvantages of this choice will be discussed later.

The word2vec embeds the input on the word level. A representation of a sentence using word2vec is the centroid (average) embedding of all the tokens in the sentence. The sentences are tokenized and stripped for stopwords and special characters. We do not use a context window for several reasons. First, sentence boundaries are not considered in the training of a word2vec model. Since we do not have any other contexts than the isolated sentences, a part of the window will be empty if the target word is close to a sentence boundary. Additionally, the relevant tokens for disambiguating the sense of the target word may not lie within the window. The sentences are picked by humans to illustrate the specific sense and they selected the amount of

[20] DanNet does contain some fixed verb phrases like *måle' op* ('measure')

context that they found relevant. We also remove stopwords as we assume they do not carry useful information for the separation of senses.

The BERT model expects a sentence as input and therefore we do not remove any stopwords, though we still remove special characters. We also add a special target token ([TGT]) around the target word to inform the model of the placement of the target word.

4.3 Static sense representations using word2vec

The challenge of using static embedding models to represent senses is the meaning conflation deficiency: how multiple, separate senses conflates into a single representation based on the lemma. One solution is to create an individual representation for each word sense. That entails a) defining all the possible senses of each lemma and b) a method of effectively distinguishing the senses with the embedding model. In this work, we generate the individual sense representations by combining word2vec and DanNet.

Each synset in DanNet is considered one sense, and therefore we have a predefined number of senses for every lemma. we use the knowledge from the synsets to differentiate the senses. We assume that the knowledge contained in a synset differs sufficiently from other synsets in the network to ensure the individuality of the representation of senses.

The information from DanNet is combined with the word2vec model by embedding every word in a synset and then calculating the average representation (the centroid). The set of words can also be considered a sense-bag, a multi-set representation of the sense. Since wordnets are constructed by experts, we can assume the synset information is the most relevant for defining that sense. By including the nodes linked to the synset (i.e., the synset members) in our sense-bag, we also follow Pustejovsky's theory of Qualia Structure (Pustejovsky J. , The generative lexicon, 1998). The members of the target synset have defined relations, and therefore can be used to determine the meaning of the target synset. This is in addition to the distributional information inherently captured by the word2vec model.

The pipeline from lemma to sense representation follows four steps (see figure 10). First, all the synsets related to a lemma are found. The information within each synset (e.g., relations, definition, example sentence) is pre-processed and added to a

sense bag. In the third step, every word in the sense-bag is embedded using the Danish word2vec model. Lastly, the mean embedding of the sense-bag is calculated and stored.

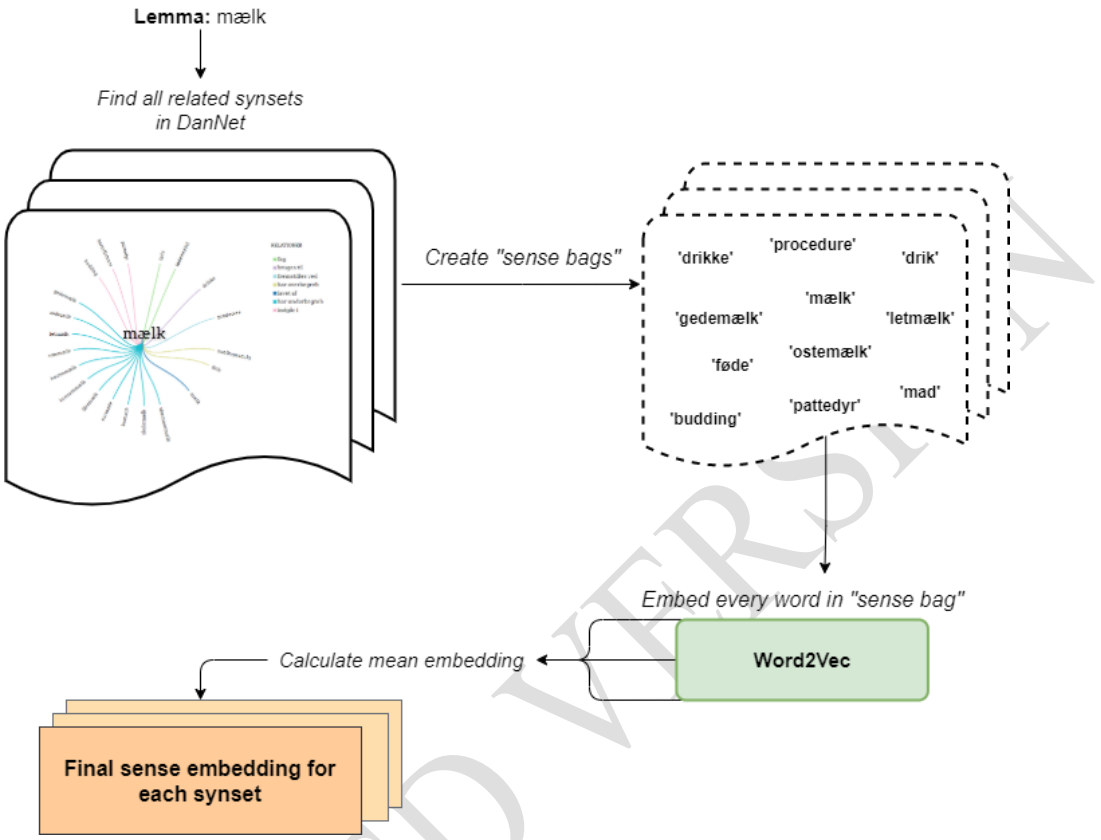


Figure 10: From lemma to sense representation pipeline using word2vec

Formally, a sense bag is considered a set, $SB = \{w_1, \dots w_n\}$, where w represents the tokens, and n is the number of tokens in the sense-bag SB. The embedding model (word2vec) is a function that maps an input token w_i to a vector representation $\overrightarrow{W_i}$. The final sense representation $\vec{S}$ can be calculated by taking the mean of every vector representation in the sense bag:

$$\vec{S} = \frac{1}{N}\sum_{i=1}^{N}\overrightarrow{W_i} \quad (1)$$

Five different types of information can be combined in the sense-bag:

1. Local synset members (directly linked nodes)
2. Local definition (max 25 characters, pre-processed)
3. Local example sentence (pre-processed)
4. Hyponym/hypernym members (from now: HH members)

5. Hyponym/hypernym members (from now: HH definition, pre-processed)

4.4 Sense representations using Danish BERT

Contextualized embedding models do not face the same challenge of conflating sense representations as do the static embedding models. This eliminates the need to actively generate separate representations for each sense, as the different senses can be invoked through the context sentence as in natural language. However, the context dependent representation does lead to the question of how we evaluate the quality of the representation of senses. In other words, what do we input to the model to get the representations of specific senses?

One method is to use a knowledge resource like a wordnet as a sense inventory as in the static embedding approach. We can then use the synset information as input to our model to invoke the different senses. The key difference is the fact that a BERT model works on sentence level, while word2vec works on word level. BERT expects a sequence of tokens as input and returns both an embedding of the entire sequence and embeddings of the individual tokens. If we follow a similar approach to the static representations, we can then input a sense-bag to the BERT model and get the sentence representation. However, the problem with that approach is that BERT is modelling sequences, and therefore the order of the input words from the sense-bag influences the final representation. Additionally, a set of words like a sense-bag differs from a tokenized sentence from natural language. This could potentially affect the sense representations by creating an unnatural semantic space.

We assess that the optimal approach with a BERT model must disregard the synset members and focus on either the definition or example sentence.

We test two versions of integrating sentences (definitions[21] or examples) with the BERT model:

1. Average of multiple sentences from DanNet. The sentences are either taken from the local synset or the local synset plus the hyponym/hypernym synsets.

[21] We are aware that the DanNet definitions are not complete.

2. A single sentence from a lexical resource (DanNet or the Danish Dictionary)

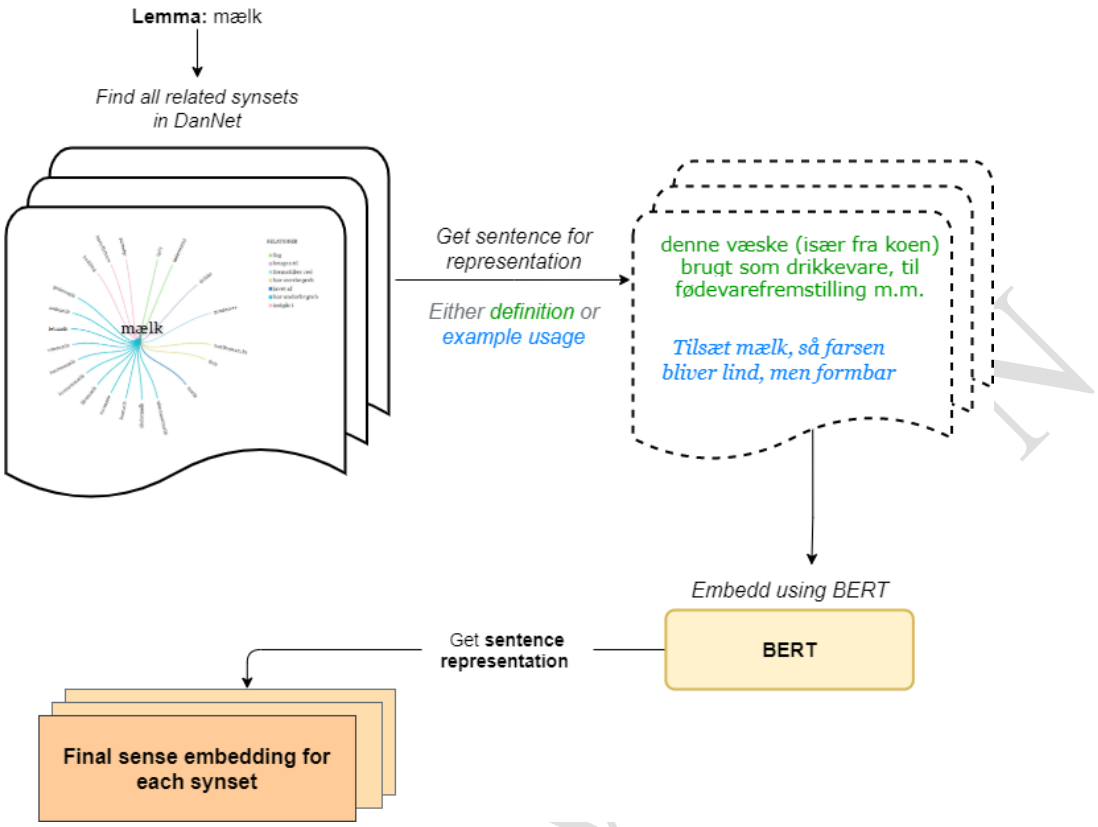


Figure 11: From lemma to sense representation using BERT

We assume that every definition or example sentence from a lexical resource is a clear example of a given sense, since they are handpicked by an expert.

When finding the sense representations related to a lemma, the pipeline follows four steps as can be seen on figure 11:

1. All the relevant synsets are found.
2. From the synsets, we retrieve either the definition or the example sentence, which will be the input sentence to the BERT model.
3. Before inputting the sentence to the model, it is lowercased, stripped of special characters, and tokenized.
4. In the third step, the pre-processed input sentence is embedded using BERT.
    - If multiple sentences are used, step 2 is repeated for every example sentence retrieved. These sentences may be retrieved from the hyponym or hypernym synsets.

5. The next step is selecting the representation type. We have the choice of using the representation for the entire input sentence (CLS token) or just the representation of the target token (the lemma).
    - If multiple sentences are used, the final representation is an average of every sentence representation.

Our experiments will use the CLS token as these are more native for the BERT model.

4.4.1 Fine-tuning the BERT

BERT models many aspects of natural language, and the model can therefore be used for many different tasks. That means that not all the information contained in a BERT embedding is crucial for our tasks of distinguishing senses.

We fine-tune the Danish BERT model by adding a binary classification layer that maps the 768-dimension last hidden layer of BERT to a layer with a single neuron. Then, we train the model with the classification layer - either the DanWiC or SemDaX task. The output from the classification layer is a prediction of how positive the label is. In our case, a positive label is when the two contexts invoke the same sense of the target lemma.

5 Evaluation

This chapter presents how the sense representations are evaluated. We will begin by introducing the intrinsic and extrinsic evaluation methods. Then, we will present two approaches for how the sense representations can be integrated with the evaluation task. We will finish by going over the setup of each experiment.

5.1 intrinsic and extrinsic evaluation

Now we know how to get a vector sense representation, from either a lemma (static, word2vec) or a sentence with a target lemma (contextualized, BERT), the next step is to evaluate the quality of the sense representations. This can be done with a focus on the internal structure and coherence of the vector space or on a supervised learning task that focusses on the real-world application. The first focus will be referred to as the intrinsic evaluation, and the latter will be called extrinsic evaluation. If possible, it is good test the representations using both intrinsic and extrinsic evaluation methods, as the performance in intrinsic tasks does not necessarily correlate with the performance in extrinsic tasks.

4.5.1 Intrinsic evaluation

An intrinsic evaluation method aims at investigating the quality and/or consistency of the properties in the representations (Camacho-Collados & Pilehvar, 2018). Examples of such properties are semantic similarity, synonymy selection, outlier detection, and sense clustering.

When selecting the intrinsic evaluation method, we focused on an approach which include the context for a word sense. Additionally, we aimed for a task that examines the conflation of senses in the same lemma. The DanWiC dataset is a newly constructed dataset for this exact purpose. It uses the word-in-context task (Pilehvar & Camacho-Collados, 2018). Instead of examining the similarity of senses in different lemmas, as is done in a sense similarity task, the word-in-context (WiC) task looks at sense similarity within the senses of the same lemma. This is done by presenting two contexts sentences that invokes the same or two different senses of the lemma. The context-pair does not get a similarity score, but a binary label that indicates the same sense (1) or different senses (0). Whereas the WSD task tests how well a model can identify the senses of a lemma given a context, the WiC task tests the consistency

with which the model can identify the senses across multiple sentences (in this case two). Since the DanWiC dataset is compiled from the DanNet example sentences, and we model the senses using those examples sentences, the task directly tests how well the different senses of the same lemma are represented, when the senses are directly compared with each other.

4.5.1 Extrinsic evaluation

An extrinsic evaluation method aims at assessing the quality of a representation when integrated in a supervised learning task (Camacho-Collados & Pilehvar, 2018). Since our focus is on preparing sense representations for sense tagging, we choose a lexical sample WSD task. Additionally, creating sense representations and a pipeline for WSD is useful for ongoing projects in Danish lexical semantics. For example, the Centre for Language Technology at the University of Copenhagen is working on sense-tagging and semantic parsing as part of the ELEXIS project[22]. we already have a labelled corpus available (SemDax) that is structured similarly to the typical WSD dataset it contains sentences with a target word and labelled DDO senses. Unlike traditional WSD tasks, the senses in SemDaX are unranked.

5.2 Integration of the sense representations

Both the intrinsic and extrinsic evaluation require that the sense representations be integrated in a pipeline for each task. We use two approaches in this work:

1. A cosine similarity-based approach (CoS)
2. A neural network-based approach (NN)

5.2.1 The cosine similarity approach (CoS)

The cosine similarity-based approach is inspired by earlier lexical semantic research in Danish (Olsen, Pedersen, & Sayeed, 2020) and semantic similarity approaches (Camacho-Collados & Pilehvar, 2018). This approach uses the cosine similarity measure to compare the distance of two vector representations, in our case a repre-

[22] https://elex.is/

sentation of a sense and a context sentence[23]. To find the sense representation that best identifies the sense in the context, we can compare the context representation with every sense representation.

The cosine similarity measure assumes that two vectors are similar if they have a similar orientation. The orientations of two vectors can be compared by calculating the angle between them. A smaller angle will then equal a higher similarity between the two vectors. When two vectors are orthogonal, the vectors are assumed the most dissimilar. Therefore, the cosine of the angle between two non-zero vectors can be used to estimate the similarity of the vectors. The measure returns a value from 0 to 1 and does not consider the lengths between the vectors. The advantage of the method is a simple implementation that does not require further training after the sense representations have been generated.

5.2.2 The neural network approach (NN)

The neural network approach tests whether the sense representations can be used as features for a learnable function. In machine learning, the learnable function is a statistical estimate of a complex function from a finite set of data (Goodfellow et al., 2016). The function is learnable if its performance on a task can improve from experience of some data. The data is a collection of examples, where each example contains a list of features (Goodfellow et al., 2016). The features are some measurements from an object or event (e.g., senses in language) that we want the function, the neural network model, to process in order to output a prediction. Therefore, the function can also be seen as a mapping from input *x* (sense representation) to output *y* (label).

In both the intrinsic and extrinsic evaluation, the objective is to estimate a type of similarity between two representation. The input is a combination of the two representations and the output is similarity score between 0 and 1. If the score is higher than 0.5, the corresponding label is 1. The combined representations are computed either as a concatenation or by inputting two sentences to the BERT. The BERT can embed two sentences at once, and therefore the CLS token is sufficient as input.

[23] With the word2vec model, the context sentence representation is the mean embedding of all the induvial words in the sentence after preprocessing.

Since the fine-tuned BERT is technically a neural network, we will also consider that a neural network method. The only difference is that the sense embedding is calculated in the same architecture that outputs the prediction.

5.3 Experiments

The following section presents five experiments with sense representations. The experiments will show us, a) to what extend our approaches to creating sense representations are meaningful, and b) the degree the sense representations can be used in a WSD task.

The first three involves the DanWiC dataset, while the final two uses the SemDaX dataset.

5.3.1 Experiment 1: CoS and DanWiC

Experiment 1 tests the synset embeddings on the DanWiC dataset using the CoS (see section 5.21). The aim is to directly test the semantic vector space by measuring the similarity between the sense embedding. The task is framed as a binary classification of whether two context sentences invoke the same sense in a target word (positive, 1) or two different senses (negative, 0).

Dataset

We use the DanWiC test set to evaluate the sense representations using the CS approach. The test set contains 1600 instances that each contain a target word, two context sentences, and a label (1, 0). Example instances from the DanWiC dataset can be seen in table 4:

Word (*w*)	**Context 1 (c_1)**	**Context 2 (c_2)**	**Label**
flamme	Midsommerens bål **flammer**	Efter maden varmede vi os ved bålets **flammer**, og senere dets gløder	0
glide	Båden **glider** af sted i det blå vand	Flyet **glider** igennem luften	1

Table 4: Examples of instances from the DanWiC dataset

We also fine-tune the BERT model on a pre-processed training set. The training set is processed to ensure balanced labels. The balanced training set is considerable smaller than the original, containing only 1020 instances compared to the original 5860.

Experimental setup

In the experiment, we compare each of the two context sentences with every sense (i.e., synset) of the target word in DanNet. The comparison is measured by cosine similarity as described in section 5.2.1. An instance is considered positive if the same sense is chosen for the two context sentences. The task does not consider whether the chosen senses correspond to the true synsets in DanNet. Both the context sentences and senses are represented a static model: word2vec (Mikolov et al., 2013; Sørensen & Nimb, 2018). We compare the 8 models among themselves and to a by-chance (0.5) baseline. These will be known as the CoS models. We do not create any BERT embeddings in this setup as they would get too bias by the data[24].

Word2vec

The representation of a DanWiC sentence using word2vec is simply the average embedding of the pre-processed sentence (described in section 4.3). We will test various combinations of the following four knowledge sources, since we have several options for what information to include from DanNet:

1. Local synset members (local)
2. Local definition (def)
3. Hyponym and hypernym members (hyp)
4. Hyponym and hypernym members (hypdef)

This totals to 8 different versions of sense embeddings. We cannot test every combination of the four knowledge sources, because some synsets in DanNet are missing a definition. The *definition* knowledge sources (def + hypdef) only appears along with the local members (local). Additionally, we exclude the example sentences from the word2vec sense embeddings considering that they introduce too much bias. The DanWiC dataset is compiled frm DanNet. The example sentences from DanNet are therefore identical to the sentences in DanWiC. Since the sense embeddings and

[24] The example sentences from DanNet we would use for the BERT are the same sentences as in DanWiC.

DanWiC sentences representations are processed in a similar manner[25], the resulting vectors will be too similar for a fair comparison[26].

Evaluation Metrics

The performance of the sense representation integrated in the CS approach is measured using the accuracy measure:

$$accuracy = \frac{number\ of\ correct}{total\ number\ of\ instances} \quad (2)$$

To understand how the models perform across the labels, we also measure the precision and recall (Jurafsky & Martin, 2018):

$$precision(l) = \frac{true\ positives}{true\ positives + false\ positives} \quad (3)$$

$$recall(l) = \frac{true\ positives}{true\ positives + false\ negatives} \quad (4)$$

Where true positives are the cases where the model predict a label *l* correctly, false positives are the cases that the model falsely predicts as *l*, and false negatives are the cases of *l* that the model misses. Precision informs us how often a model is correct, when it predicts a label, while recall tells us good the model is at detecting a label.

5.3.2 Experiment 2: DanWiC and the neural network method

Experiment 2 tests the performance of neural network models that uses the sense embeddings as input on the DanWiC task. The purpose is a) to test how well we can train a model to predict the labels from the sense representations, and b) test a BERT on the DanWiC data.

The task is the same as in experiment 1 with one change. Instead of first selecting a sense for each context sentence and then comparing the selected senses, we attempt to predict the label from the sense embeddings directly. It is possible that the best

[25] By processed, we mean that they are both the mean embedding of all the tokens in either the sense-bag (sense embedding) or sentence (sentence embedding). Adding the example sentence to the sense-bag would mean that the two embeddings will share a large portion of information.

comparison between the embeddings is not the cosine similarity measure, but a learnable function (the neural network). The neural network can learn which weights to give the individual features, and in that way extract the relevant information in the embeddings for sense distinctions. Additionally, we can escape the problem of the example sentences from DanNet, since we do not explicitly model the information from DanNet. Instead, we assume that the context sentences alone can be used as representations for senses. We view them as a source of sense-related information similar to the DanNet synset information.

Dataset

In this experiment, we use all three subsets of DanWiC. That includes the balanced training set with 1020 instances, the development set with 800 instances, and the test set with 1600 instances.

Experimental setup

The experiment is structured as a typical machine learning experiment with a training phase and a testing phase. We use a sentence similarity objective, where we assume a sentence representation can be used as an estimation of a given sense. In total, we will compare three models:

	Name	Description
1.	Word2vec NN	Word2vec embeddings in a neural network trained on the training set.
2.	BERT NN	Fine-tuned BERT embeddings used in a separate neural network trained on the development set.
4.	BERT FT	BERT model fine-tuned on the training set.

Table 5: Models tested in experiment 2

A comparison between the word2vec and BERT neural network can tell us if we gain anything with a deeper and computational stronger model. The intention behind the

two BERT models is to investigate how the BERT embeddings perform outside the fine-tuning setup. This can show whether the BERT embedding can be used separately from the model[27].

Word2vec NN

In the neural network model, we are not using the sense-bag representations as in the CS approach. Instead, we want to see if the word2vec representations of the two context sentences alone are enough to distinguish the senses of the target word. This is essentially almost the same as a sense-bag representation using only the synset example sentences from DanNet. The differences are (a) we do not know the number of senses for each target word, and (b) we have only one example sentence, where some synsets in DanNet have several.

The two context sentence representations are concatenated before feeding them into a feed-forward neural network model. The context representations are calculated as in the cosine similarity experiment by taking the average embedding of the tokens in the sentence without stopwords and special characters.

The neural network model is a fully connected feed-forward neural network with two hidden layers (50 neurons, Tanh activation: 160 neurons, ReLU activation). The first layer acts as a bottleneck to extract the most relevant information, which is further processed by the second layer. The architecture is chosen from initial testing with the development set. It is trained with a learning rate of 0.001, mean square error loss, batch size 16, over 25 epochs.

BERT FT

The BERT FT is the fined-tuned BERT model. The fine-tuning of BERT on the DanWiC task requires adding a fully connected linear layer with a single neuron on top of the pretrained BERT. The input to the linear layer is the pooled output of the classification token. That means the CLS token has been further processed by a linear layer with a tanh activation function, with weights that have been set during pretraining. When we fine-tune BERT, we update the weights in the entire model. The

[27] We might want a setup where we save the BERT embeddings and reuse them in another system.

BERT model is fine-tuned by using a learning rate of 2e-5, batch size of 1, max sequence length of 256, over 15000 training steps (repeated the 1020 training instances).

BERT NN

The BERT NN model is a fully connected feed-forward neural network that uses BERT embeddings as input. Since BERT allows two sentences as input for a sentence classification task, we can use the representation of the classification token as a representation of both sentences. The input to the neural network is therefore a 768-dimension embedding. The BERT model that we use for generating the embeddings have been fine-tuned on the training set before being fed into the neural network.
The neural network has the same architecture and hyperparameters as the word2vec NN model besides the different sized input layer (768 to match the BERT embedding). The training of the BERT NN only updates the weights of the feed forward network and does not influence the weights of BERT.

Evaluation Metrics

The performance of the neural network models is measured using the accuracy, precision, and recall measures. All scores are an average of 5 runs of the models. We report the accuracy for both the training and test set, so we also have an indication of how well the models are trained.

5.3.3 Experiment 3: DanWiC comparison with the human responses

Experiment 3 compares the performance of the models to the human participants from the DanWiC project (Hau Sørensen, 2021). The aim is to get an idea of how close the models are to human performance. The accuracy of the human participants suggest how well we can expect the models to perform: a ceiling level.
The task is the same as in experiment 1 and 2, but on a smaller subset of the test set.

Dataset

The dataset consists of 200 instances retrieved from the DanWiC test set. Each instance is rated by 3-4 human participants. Half of the dataset contains clustered sens-

es. As we only work with the unclustered senses[28] in DanWiC, and we have access to the true unclustered labels, we have transformed the 100 clustered instances to match the unclustered labels. This means that the dataset is unbalanced. There is an over-weight (119 vs. 81) of negative instances (different senses of the target word).

Participants

The responses from 8 human participants are used. Their ages range between 22-75 and the mean age is 41. The participants are either from the Center of Language Technology (CST) or the linguistics programme at the University of Copenhagen, ensuring the participants have some knowledge of semantics by either having taken a course in semantics or worked in the field of computational semantics.

The data has been collected by an online survey, where participants in principle had no access to other knowledge about the senses than their mental lexicon and the two context sentences.

Models

The following four models from the previous experiments are used:

	Name	**Description**
1.	Word2vec Cos3	Word2vec sense-bag embeddings using three knowledge sources (local, def, hypdef) (1st place in experiment 1)
2.	Word2vec Cos4	Word2vec sense-bag embeddings using four knowledge sources (local, def, hyp, hypdef) (2nd place in experiment 1)
3.	Word2vec NN	Word2vec embeddings in a neural network (2nd in experiment 2)
4.	BERT FT	BERT fine-tuned on the training set. (1st in experiment 2)

Table 6: Models tested in experiment 3

[28] And the clustering was unsuccessful.

Evaluation metrics

We compare the predictions of each model with the human labels on the following metrics:

- **Accuracy** to test the overall performance on the dataset.
- **Model-human agreement** by computing an average Cohen Kappa score between the model and all human participants.
- **Model-labels agreement** by computing the Cohen Kappa score between the model and the true labels.

The Cohen's Kappa is an agreement measure that controls for change agreement (Cohen, 1960). It is measured by:

$$Cohen's\ Kappa = \frac{p_o - p_e}{1 - p_e} \quad (5)$$

Where p_o is the agreement between two annotators (in percentage) and p_e is the expected percentage of agreement (Cohen, 1960).

We additionally present two set of words from the dataset:

1. The lowest scoring words for all models that has high accuracies in the human participants.
2. The 10 lowest scoring words for each model that also has low accuracies in the human participants.

5.3.4 Experiment 4: SemDaX and the cosine similarity method

Experiment 4 is the first of two WSD experiments on a subset of SemDaX. The experiment aims at testing whether the semantic space created by the sense representation can be directly used for sense-tagging of Danish nouns.

Dataset

In the experiment, we use a subcorpus of the semDaX dataset that is annotated with senses from DDO. The data contains 5977 instances with 17 highly ambiguous Danish nouns. Each instance contains a context sentence, a target lemma, and one or more sense labels. The distributions of senses and instances for the target lemmas can be seen in section 3.2.2. Table 7 shows two examples from the SemDaX dataset.

Word (*w*)	Context	Label
ansigt	Jeg har læst et eller andet sted, at man bruger 47 muskler i **ansigtet** på at se sur ud, men man bruger kun tre på at se glad ud.	ansigt-1
skade	Til stor **skade** for de studerende og uddannelserne	skade-1-1c

Table 7: Example of two SemDaX instance

The sense representations are based on DanNet senses. Therefore, we use a mapping from DDO to DanNet created by Olsen (2020).

Experimental setup

The task is a multilabel classification problem, where the number of classes depends on the number of senses for each lemma. We compare a context representation to every possible sense of the target lemma. If the selected sense matches a labelled sense, the instance is considered correct. Since some instances can have multiple labels due to disagreement in the annotators, the selected sense need only match one of the annotated labels. The context sentence is represented by the same embedding model that is used to create the sense representations.

In the experiment, we use different versions of the static and contextualized representations. We use the most frequent sense as a baseline.

Word2vec

The representation of a context sentence using word2vec is the average embedding of the preprocessed sentence.

The word2vec sense representations are static representations as described in section 4.3. Therefore, each sense representation is the average embedding of a sense-bag extracted from DanNet. Unlike in the DanWiC experiments, we can safely include the example sentences in our sense representations in this experiment. Therefore, we will test various combinations of the following five knowledge sources:

1. Local synset members (local)
2. Local definition (def)
3. Local example sentence (exam)
4. Hyponym and hypernym members (hyp)
5. Hyponym and hypernym members (hypdef)

BERT

The context sentence using BERT is simply the CLS token representation of the sentence.

The BERT sense representation is the average CLS token representation of a list of sentences retrieved from DanNet. The sentences are retrieved from either the local synset (definition or example sentence), the hyponym synset, or the hypernym synset. In total, we create five different version of sense representations using BERT:

1. Local example sentences (local)
2. Hyponym example sentences (hypo)
3. Hypernym example sentences (hyper)
4. Combination of hyponym and hypernym example sentences (hypo/hyper)
5. Definition (def)

All the versions use a BERT that is fine-tuned to the DanWiC balanced training set. Since we do not have the same problem with the DanNet example sentences in this experiment, we can compare the performance of the contextualized sense representations with the static sense representations.

Evaluation metric

We use the accuracy measure to evaluate the performance of the sense embeddings. Since some instances contain multiple labels, those instances are considered correct if the prediction matches one of the labels. We do not measure the precision and recall as instances contain varying number of classes.

5.3.5 Experiment 5: SemDax and the neural network method.

Experiment 5 has the purpose of investigating whether we can solve the WSD task by training a neural network from the embedded SemDaX context sentences. Like experiment 4, the experiment is a multi-label classification task with a varying number of labels. This requires adapting to the neural network architecture, which we do through a sense-selection objective.

Sense-selection objective

In a typical multi-label task, we would have one neuron in the final layer for each class. Since the number of classes depends on the number of senses of the target lemma, one could solve the problem by training different networks for each target. However, that method is not scalable. Instead, we are inspired by the objective that is used to fine-tune GlossBERT (Huang et al. 2019) and further developed by Yap et al.

(2020). They use a Gloss Selection objective where positive and negative pairs of context sentence and sense definitions are constructed from a sense-annotated dataset and a wordnet (Fellbaum, 1998). A positive pair has the definition of the labelled sense, while a negative pair has the definition of another sense. Likewise, we pair every potential sense (i.e., every sense listed for the target lemma in the lexical resource) with the same context sentence. The context-sense[29] pairs that corresponds to an annotated sense get a positive label (1) and every other target-sense pair gets a negative label (0).

Each instance in the SemDaX dataset is still considered one training example, however the training example is now a mini batch of the context sentence matched with every possible sense. We then input an entire mini batch to the neural network model. The output of the model is a relevance score of how relevant a sense is to the context, similar to how a web document can be compared to a query in information retrieval. Therefore, we only need a single output neuron in the neural network, no matter how many senses a target lemma has.

During training, the output from a mini batch is a list of relevance scores, which are treated like activations from a multi-neuron output layer. The index of the correct label is used as the label in a cross-entropy loss function. To handle the possibility of multiple labels from disagreement between the annotators, we add a second binary loss function. Thus, the task also used multitask learning: one task is to select the correct sense, and the other task to estimate whether a context-sense pair is a match. The combined loss function for a training instance is:

$$loss = -\sum_{i=1}^{n} 1(y,i)\log(p_i) + \frac{1}{n}\sum_{i=1}^{n}(1(y,i) - Rel(c,s_i))^2 \quad (6)$$

Where n is the size of the mini batch, $Rel(c,s)$ is the relevance score between context c and sense s_i, $1(y,i)$ is a binary indicator of index i matches the index y of a

[29] We call our pairs for context-sense pairs as we do not limit ourselves to the gloss from a wordnet.

positive context-sense pair, and p_i is the softmax value of the i-th relevance score. The softmax value is calculated as:

$$softmax(Rel(c, s_i)) = \frac{\exp\big(Rel(c, s_i)\big)}{\sum_k^n \exp\big(Rel(c, s_{ik})\big)} \quad (7)$$

This approach has the possibility to rank all the senses in a mini batch. We will however only select the top-1 sense.

Preparing the data

Since SemDaX is annotated with DDO senses, we will primarily base this method on that sense inventory. This will allow us to handle fixed phrases by adding their senses to the mini batch candidates. To save memory on the GPU, we only add the example sentences for senses that are present in the data.

We additionally make a definition and a combined example-definition dataset to compare which sentence type is best for modelling senses. Both the definitions and example sentences were manually extracted from the online version of DDO. Since the definitions do not necessarily contain the target lemma, we add the lemma to the start of the definition in a phrase, for example *X is a + definition.*

Experimental setup

The experiment contains two tests. The first test contains a comparison of two word2vec NNs and a BERT FT trained/fine-tuned on the sense-selection objective. The two word2vec NNs are trained on DDO and DanNet senses, respectively.

The second test evaluates which sentence type (definition, example sentence, combined) is best for the sense-selection objective.

Since we need training data in this task, we split and rotate the dataset in five folds (80/20 training-test split) using stratified cross-correlation. This rotation will ensure that we test the models on every instance of the data at some point, which will give us a better view of their generalisation abilities.

Word2vec

We train two word2vec NN models that are based on different sense inventories. The first model is based on DDO senses. It treats the context-sense pair like the two context sentences in experiment 2. Each sentence (context or example sentence) is the centroid of the token embeddings from the pre-processed sentence. The two sentence

representations are then concatenated and feed into a feed-forward neural network. This model is called word2vec DDO.

The other version is based on DanNet senses and uses the best sense representation model of experiment 4 (from now, top CoS). It represents the context sentence like the first model; however, the sense sentence representation is replaced by the sense representation from top CoS. The context and sense representations are then concatenated and fed into the neural network. This model is called word2vec Synset.

The two NN models use the same architecture as in experiment 2.

BERT

We fine-tune a BERT on either a dataset with example sentences only, definitions only, or both combined. The number of training instances are fixed to 15000, regardless of the size of the dataset. This model is re-initialised before every fold in the cross-validation.

Evaluation metric

Like in experiment 1 and 2, we use the accuracy measure to evaluate the performance of the sense embeddings. Since some instances contain multiple labels, those instances are considered correct if the prediction matches one of the labels.

The neural network model is a fully connected feed-forward neural network with two hidden layers (50 neurons, Tanh activation: 160 neurons, ReLU activation). This is the same architecture as in experiment 2. It is trained with a learning rate of 0.001, batch size 1, over 15000 training steps.

6 Results

This chapter presents and analyses the results from experiment 1-5.

6.1 Intrinsic evaluation

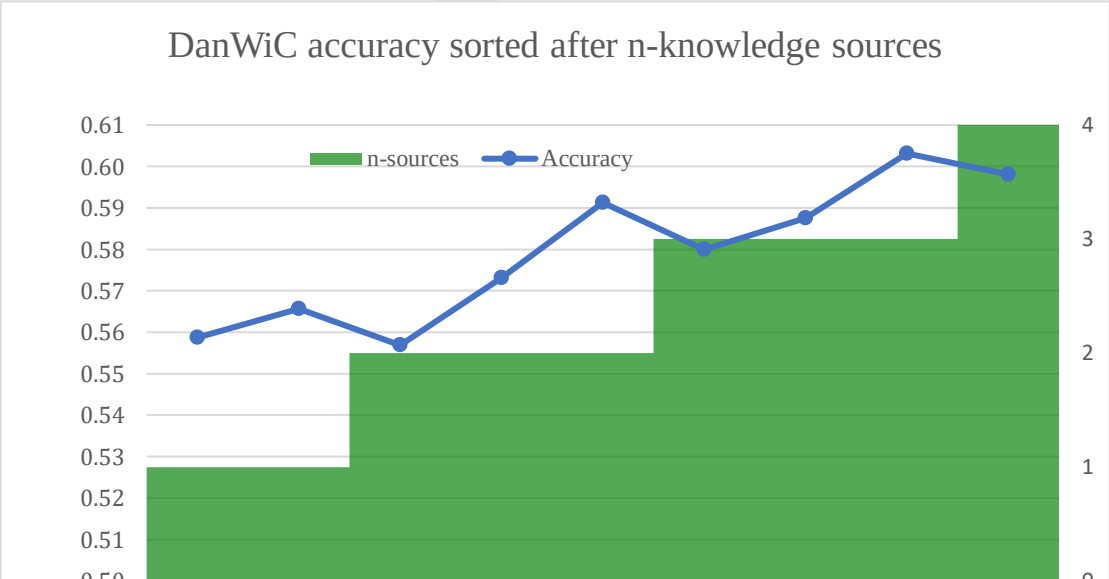
The intrinsic evaluation tests the quality of the sense embeddings, both by measuring distances between embeddings (CoS approach) and in a learning setting (NN approach).

6.1.1 Experiment 1

Experiment 1 evaluates the semantic vector space by directly measuring distances between embedded sense representations. The semantic space is meaningful, if it can accurately group similar senses (i.e., examples from the same synset) and distinguish separate senses (i.e., examples from different synsets).

Cosine Similarity with Static Representations

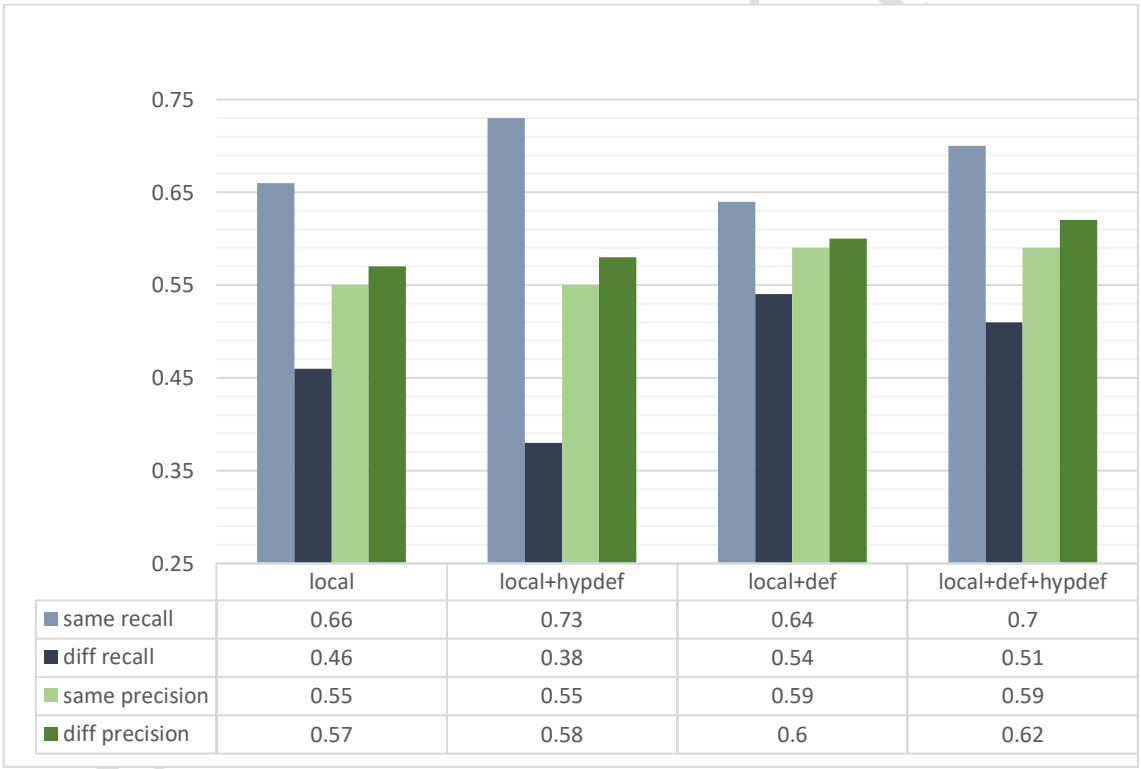
Figure 12 shows the results on the DanWiC test set for 8 CoS models. Models achieve higher than a random baseline (0.5). The highest accuracy is achieved by the combination of local members, local definition and HH definition (0.6).



	local	hyp	local + hypdef	local + hyp	local + def	local + hyp + hypdef	local + def + hyp	local + def + hypdef	local + def + hyp + hypdef
n-sources	1	1	2	2	2	3	3	3	4
Accuracy	0.559	0.566	0.557	0.573	0.591	0.580	0.588	0.603	0.598

Figure 12: Accuracies (blue) for the 8 versions of the word2vec sense embeddings on the DanWiC task sorted after number of knowledge sources used in the sense embeddings (green).

The knowledge sources affect the accuracy to different degrees. Alone, the HH members (hyp) perform better than the local members (local), but they negatively affect the accuracy in combination with the local definition. The local definition, on the other hand, increase the accuracy the most when combined with other knowledge sources[30]. This is especially true in combination with HH definitions. One thing to note is the fact that the synset knowledge from HH members contain drastically more tokens than any other knowledge source. On average across all synsets in the test set, the HH members alone contribute with 353 tokens, while the combination of all the other knowledge source only contains 29 tokens. This is inefficient, since HH definitions can represent similar information about the taxonomies of the senses but with fewer words. Additionally, the high number of tokens also results in proportionally lower weights for the other knowledges sources in the sense embeddings.



	local	local+hypdef	local+def	local+def+hypdef
same recall	0.66	0.73	0.64	0.7
diff recall	0.46	0.38	0.54	0.51
same precision	0.55	0.55	0.59	0.59
diff precision	0.57	0.58	0.6	0.62

Figure 13: Precision (green) and recall (grey) for four static sense embedding models.

[30] We could not evaluate the definition alone as not every synset has a definition. This applies to the HH definition too.

To understand how the local definition and HH definition influence the sense embeddings, we examine the precision and recall for each label (same sense, different senses) on figure 13. The knowledge sources are most distinct on the recall measure. The HH definitions are effective at detecting the same senses (light grey) but have difficulty with distinguishing senses. The local definition contributes to a higher recall of the different senses (dark grey), which indicates that it is better at separating senses. It also achieves a higher precision. Combined, the two definition types supplement each other well in that they balance the recall for the two labels. To improve the best scoring model (local+def+hypdef), the recognition of different senses must improve, as the model is biased towards the same sense label.

6.1.2 Experiment 2

Experiment 2 evaluates the sense embeddings performance as feature vectors for three different neural network models: word2vec NN, BERT NN, BERT FT. We first report the accuracies for all models, and then the precision and recall of the best two.

Accuracy

The accuracy scores for experiment 2 can be seen in table 8. On the test set, the BERT FT achieves the highest accuracy of 0.59. This is a similar performance to the best models from experiment 1.

	Training	Development	Test
Word2vec NN	*0.99*	0.54	0.53
BERT NN	0.48	*0.51*	0.49
BERT FT	*0.99*	0.61	0.59

Table 8: Accuracies for the neural networks on the DanWiC training, development, and test sets. Italics indicate which subset is used for training the models.

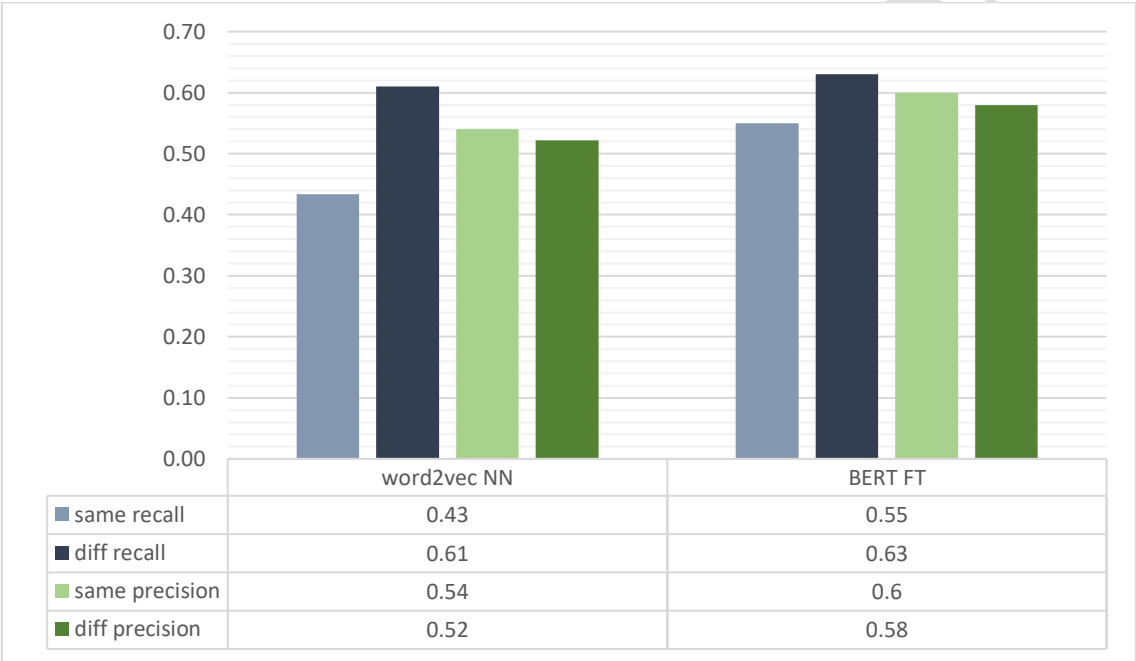
The word2vec NN just outperforms the random baseline of 0.5. The high difference between the accuracy on the training and the other sets indicates that the model overfits to the training data. Therefore, it is not able to generalise to neither the development nor test set.

Though a fine-tuned BERT model was used to generate the input embeddings to the BERT NN, the model still fails at beat a random baseline. The BERT NN's accuracy on the development set also shows that the model is unable to learn from the contex-

tualised embeddings. This indicates that the BERT embeddings may not be meaningful outside of the BERT framework.

Precision and recall

The precision and recall for both labels (same and different senses) are seen in figure 14. The NN models shows the opposite tendency from the CoS models in that the recall for the different sense label is the highest. The word2vec has a similar different sense recall as the BERT model, though the precision is at the same time lower. The higher recall stems from the fact that it simply classifies more cases as different senses.



	word2vec NN	BERT FT
same recall	0.43	0.55
diff recall	0.61	0.63
same precision	0.54	0.6
diff precision	0.52	0.58

Figure 14: Precision (green) and recall (grey) for the NN models on the DanWic test set for the two labels (same and different senses).

6.1.3 Experiment 3

Experiment 3 is a direct comparison of the best models with human participants. The comparison can enlighten:

a) How close we currently are to the ceiling performance determined by humans (measured by accuracy and performance).

b) Whether the models fail at capturing semantic properties that are not difficult for humans. This can tell us which areas of lexical semantics are especially hard to model from word distributions and how to improve in future work.

c) Whether the sense inventory is too fine-grained in some areas for the distinction to be reflected in the mental lexicon of the human responders and distributionally in the embedding models.

The results for experiment 3 are split into four sections. The first two attempts to answer (a), the third will look into (b), while the last will investigate (c).

Accuracy

Table 9 shows accuracies for the models and the average of the human participants. None of the models come close to the human accuracy of 0.72. This shows that we still have a gap to close with the sense embedding models. The best model is the best static embedding from experiment 1 (word2vec CoS-3). The BERT FT comes second, however this accuracy is an average of all five runs. One run was slightly worse than the others and therefore negatively influenced the average accuracy. The best run of the BERT FT has an accuracy of 65.5%, though it is still not good enough to reach the human ceiling.

	Human	word2vec CoS-3	word2vec CoS-4	word2vec NN	BERT FT
Accuracy	**0.72**	0.64	0.60	0.58	0.63

Table 9: Accuracies for the models and human participants (avg)

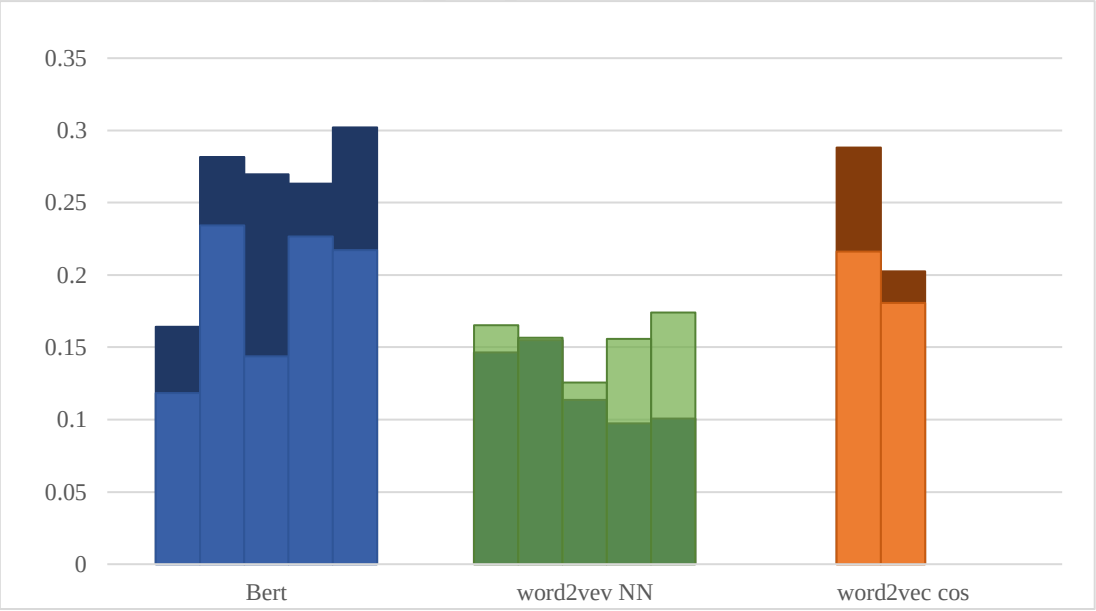


Figure 15: The Cohen Kappa's score for model-to-human agreement (light) and mode-to-label agreement (dark). The model-to-human agreement is an average of the agreement score between a model and every human participant. Each column is a separate run of the model.

Agreement

Figure 15 shows the Cohen Kappa agreement score for each run of the models. The light columns are the average score between a model and every participant, while the dark columns are the score between the model and the true labels. All the models have higher than change agreement with the human participants and true labels (a value higher than zero). The word2vec coS3 (orange, right) and the three best runs of BERT FT get a fair agreement with the human participants (0.21-0.4). The word2vec NN fail to get a better than slight agreement with both the humans and the labels (0.0-0.2). In fact, it appears that the word2vec NN agrees more with the human participants than the labels. This indicates that the word2vec NN gets similar instances wrong as the humans.

The model-to-human agreement is lower than the human-to-human agreement (0.35-0.5) (Hau Sørensen, 2021). The model-to-label agreement (<0.3) is also lower than the human-to-label agreement (average of 0.32). This shows that the models do not attain the same sense distinctions as humans. The best performing models do get close to the human-to-label agreement while still achieving lower agreement with the humans. We can learn two things from this:

a) A low model-to-human agreement and a high model-to-label agreement shows that the models get instances correct that the humans do not.
b) The difference between the model-to-label and human-to-label agreement can be contributed to the instances that the humans get correct, but the models do not.

Lowest scoring words with high human accuracies

Table 10 shows the target words of the instances that have a perfect score in the human participant, but a low accuracy in the models. To explain the mistakes, we can split the instances up in three groups:

1) Related to polysemy (green), possibly non-linear polysemy (blue).
2) Related to the topic/domain of the context sentences (orange).
3) Related to word classes, errors/unclarity in DanNet, or other (red).

We will now go over some of the examples from each group. We will mainly focus on the CoS models, as we know exactly what semantic knowledge is used in the modelling.

	Word2vec Cos	Word2vec NN	BERT
Incorrect in all models	marginal (1) grund (1)	marginal (1) grund (1)	marginal (1) grund (1)
Incorrect in two out of three models	sympati (0) indie (1) bearbejde (0)	indie (1) bearbejde (0) klokke (0) sammenstød (1) prøve (0) passe (0)	Sympati (0) klokke (0) sammenstød (1) prøve (0) passe (0)
Incorrect in one model	stram (0) tydelig (0) camouflere (0) guldklump (0) betrække (0) forsvarsposition (0) hvile (1) rest (1)	drøm (0) ballondans (1) krukke (1) skyde (0) stor (1) skæg (0) skær (1) udsætte (1) kradse (1)	refleks (0) kende (0) sprøjte (1) rækkefølge (1) brød (0) gungre (1)

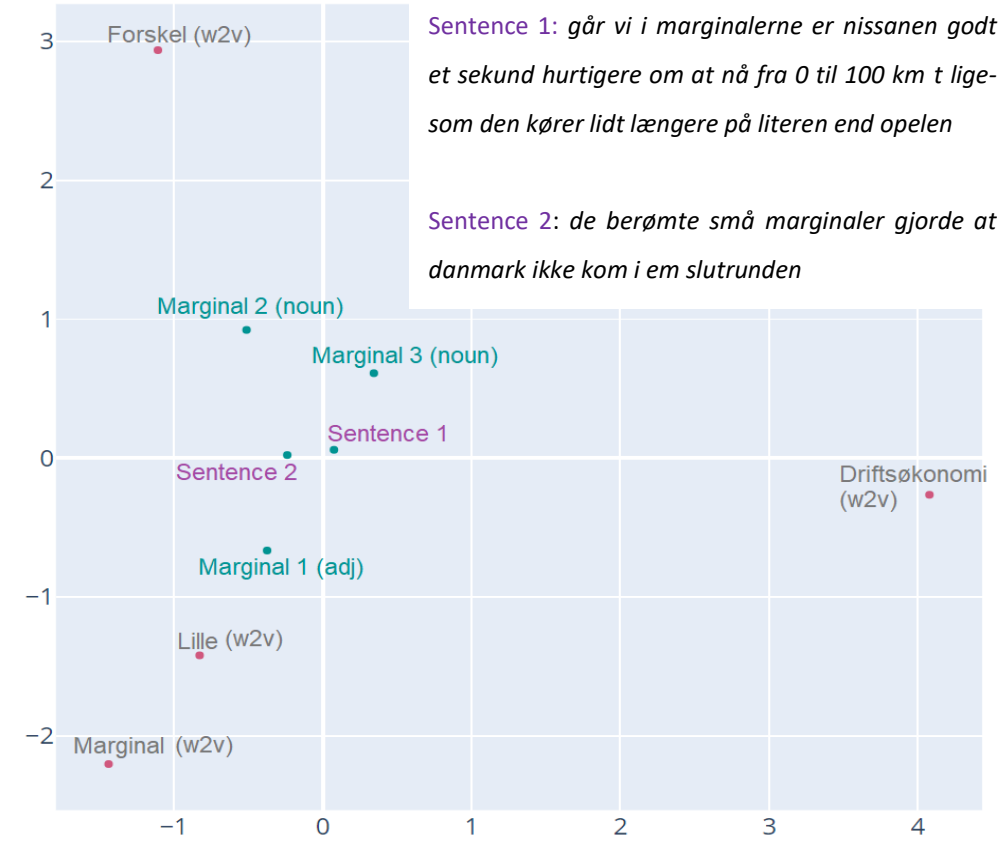
Table 10: Target lemmas that the models got wrong but the human participants did not.

All the models misclassify the instance with the target word *marginal* ('small, marginal') which has a "same sense" label. The CoS models assign the correct sense to the first context, however the second context is either assigned the more specific sense of 'margins in economics' (CoS3) or the adjective sense 'small' (Cos4). We have three explanations:

1. The synset information is not sufficient to separate the three senses. This could be from overlap in synsets caused by shared relations (both noun senses have the same hypernym *forskel* 'difference') or similar word in the definitions (all three synset definitions contain the word *lille* 'small').
2. The synset embeddings on may contain tokens from multiple word classes. Therefore, the representations are not sensitive to word classes.

3. The second context sentence may contain tokens that are not directly important to determine the sense of the target word. These tokens will move the context embedding away from all the sense embeddings, which makes it more difficult to select the correct sense.

We can investigate the possible explanations through a visualisation. On figure 16 we can see that the context sentences are in close proximity to each other. The two noun senses are also close. In this case, it is most probable that the two first explanations caused the mistakes by the models.



Marginal 1	marginal, lille, ubetydelig, fylder, rumlig, fysisk, ringe
Marginal 2	marginal, forskel, talværdier, afstand, forskel, størrelsen, mængder, adskiller
Marginal 3	marginal, forskel, virksomhedsøkonomi, driftsøkonomi, indtægter, udgifter, forskel, størrelsen, mængder, adskiller

Figure 16: 2D visualisation of the semantic vector space using word2vec Cos3 and PCA for dimensionality reduction. The red markers are single word representations, while blue markers are the average representation of a list of words (sense-bag or sentence). The table shows the words used for the sense-bag representations.

To investigate the neural network models, we can also visualise their representations. Since the BERT FT makes a combined representation of both sentences, the visualisation model is not the same as the one that classified the instances. Instead, we make separate sentence embeddings. We also add embeddings of the three senses and their hypernyms calculated as the centroid example sentences for every synset. This can be seen in figure 17.

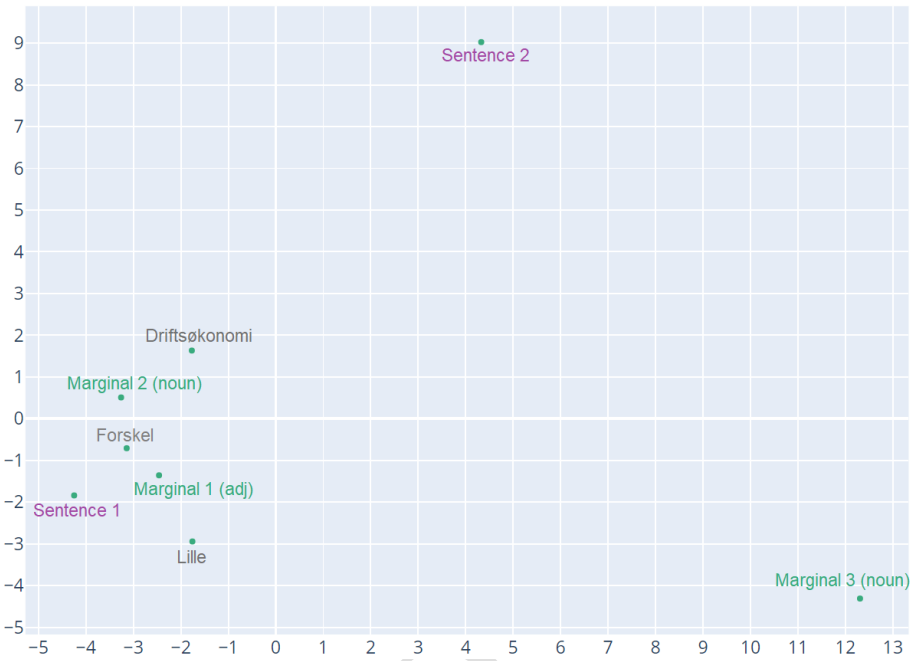


Figure 17: 2D visualisation of the semantic vector space using a fined-tuned BERT with separate embeddings for the two context sentences (purple). The grey words are the hypernyms to each sense.

Unlike with the cosine-similarity based models, we see a clear distinction between the two noun senses of *marginal*. The two context sentences are far apart, and surprisingly sentence 2 is also far from *marginal 2* (average embedding of both sentences). This may be caused by the third explanation from above. The distance between the embeddings can cause problems, as neither word2cec NN or BERT FT explicit represent the sense as in the sense-bag representations. Instead, the task is framed as a sentence similarity task. Therefore, the models may falsely separate the contexts because of different topics expressed in the sentences. This could especially influence the word2vec NN that operates without any marking of the target and syntactic information.

This can explain the mistakes with the case of *sammenstød* ('collision, clash'), where the models ascertain that the following two contexts contains different senses:

(2a) I Indien har der været alvorlige **sammenstød** mellem hinduer og muslimer i 12 delstater.

*In India, there have been serious **clashes** between Hindus and Muslims in 12 states.*

(2b) Efter et større **sammenstød** med min far, pakkede jeg min ransel og stak af hjemmefra.

*After a major **clash** with my father, I packed my knapsack and ran away from home.*

The two contexts have the same intended sense, which is 'a violent or unpleasant quarrel or conflict between two parties'[31]. Since this is a metaphorical extension of the literal sense of *sammenstød* ('forceful crash or collision'), one could think the models misclassify the example because they cannot differentiate literal and figurative senses. In table 11, we see the similarity score that the NN models give to the context sentences from both the literal and figurative senses.

Word2vec	crash	clash
crash	0.999	0.994
clash	0.999	0.0

BERT	crash	clash
crash	0.872	0.077
clash	0.284	0.11

Table 11: Output activation (similarity score) from the models on the 'sammenstød' instance.

We see that both models can correctly identify the literal sense 'crash' in two context sentences. The BERT model can also distinguish the figurative from the literal sense, while the word2vec does not. However, if the word2vec did not distinguish between the figurative and literal sense, we would expect the model to classify all the senses as the same. Instead, we can argue that tokens like *indien* (india), *hinduer* (Hindus), and *delstater* (states) are far from tokens like *far* (father), *ransel* (knapsack), and *hjemmefra* (from home). The sentence representations become too dissimilar to generalise the similarities in the events (two parties having a conflict).

[31] https://ordnet.dk/ddo/ordbog?query=sammenst%C3%B8d

This can be further supported with the *ballondans* ('balloon dance') instance. The lemma is not ambiguous, so the sense cannot be confused or conflated with other senses. The word2vec NN does however misclassify the instance, which shows how sensitive the model is to different sentence topics. This is a problem that can arise when working with sentence representations. We cannot ensure that the model learns to differentiate senses and not some other features in the context sentences (e.g., topic or domain). Another example of problems with sentence representations is the instances with *prøve* ('test', 'try) and *kende* ('a bit', 'know'). The word class of the target words are different in the two context sentences (noun vs. verb). However, the neural models still fail to distinguish the senses.

Besides the problems of topic and word class, we expect the remaining instances from table 10 are related to some degree of non-linear polysemy. This is primarily because it is easy for humans to detect figurative senses compared to the fine distinctions created by linear polysemy. The instances marked with blue can all be linked to non-linear polysemy without an obvious alternative explanation. This is the case with *grund* that all the models misclassify. The intended sense of both context sentences is the metaphoric extension of the literal sense ('foundation') and the figurative sense ('motivation'). The BERT model does not distinguish between the literal and figurative sense, when comparing two literal context sentences with the two figurative. The word2vec separates all contexts sentences with *grund*, and therefore does not correctly detect when the literal sense is used in both sentences. The CoS models correctly identify the figurative sense in sentence (3a), yet both assign the literal sense in sentence (3b):

(3a) Jeg holder på at man altid skal sige sandheden, for det første af moralske **grunde**, for det andet af praktiske **grunde.**

*I stick to the fact that you should always tell the truth, first for the moral **reasons**, secondly for practical **reasons**.*

(3b) Hvad er **grunden** til at du har valgt tysk som fag?

*What is the **reason** why you have chosen German as a subject?*

We contribute this to the few words in the second sentence and the use of the static word2vec model. The different senses of *grund* are still conflated in the token embeddings in both the sense and context representations. The figurative sense may be distinct in the sense embedding, but if a sentence is short or contain many stopwords,

the other tokens in the sentence are not enough to pull the centroid context embedding towards the correct sense. The four other senses of *grund* in DanNet are all literal senses (related through linear polysemy). Therefore, it is likely that literal senses are overrepresented relative to the figurative sense, and therefore the lack of proper context results in a literal prediction.

Low-performance instances across models and humans

The target words of the instances with low accuracies for both the models and human participants are seen in table 12. The sense distinctions in these examples are related to fine-grained distinction, typically linear polysemy with the same hypernym.

Word2vec Cos	**Word2vec NN**	**BERT**
internet	**internet**	tynde ud
græsstrå	te	græsstrå
te	tynde ud	flakse
våbenfælle	affotografering	vende
flakse	vende	naturalistisk
naturalistisk	omløbstid	**rund**
affotografering	uendelig	knække
rund	tilbageholde	**internet**
tværsnit	**rund**	rykke
opholdsvejr	tværsnit	forøgelse

Table 12: Target lemmas that both the models and humans got incorrect.

For instance, all the models have trouble distinguishing *internet*, which is split into a sense for 'the world wide web of computer connections' and the subsense of 'the facility on the web'. The distinction between the technical and facility interpretation is expressed through the propositions *over* 'over' and *på* 'on'. This information is absent in the CoS sense embeddings since we have removed the stopwords. However, this does not explain why the human and NN models have trouble detecting the senses. The technical sense builds on how the internet functions, while the facility sense builds on how it is used. As most humans mainly interact with the internet as a facility, it is questionable whether the technical perspective can be derived from the language use. It might require modelling of additional real-world knowledge to cap-

ture the sense, though that may be an unnecessary step as even the human participants did not make the distinction.

Another example is the adjective *rund* ('round'). The three listed senses in DanNet refers to different properties of round shapes (circle, sphere, cylinder). However, they all have the same single relation: the hypernym *form* 'shape'. They only differ in the definition and example sentence. One could argue that these properties are not distinct as they all refer to a concept of 'something unsharp without edges'. The specific property is linked to the noun head that the adjective modifies. Following the mechanism of type coercion in Pustejovsky's generative lexicon (Pustejovsky J. , 1995), the different interpretations are semantic transformations of the entities that binds to a specific subpart of the meaning of the noun. When we talk about a *round stomach* and a *round plate*, the *roundness* property transforms depending on constraints in the meaning of stomach and plate, respectively. Therefore, we might only need one sense that can naturally adapt to the different contexts. Nevertheless, with such limited knowledge from the wordnet, it is impossible to make distinct enough representations.

Another example is *græsstrå* 'straw of grass'. The senses refer to different parts of the plant ('stem' versus 'straw, leaf'). However, the synsets in DanNet are again almost identical. The only difference is the hypernym (*stængel* 'stem' vs. *strå* 'straw'). However, the hypernym of one sense (*stængel*) is the parent hypernym of the other sense's (*strå*). This is a fine distinction that is not useful outside the scope of botany, and one can ask how important this distinction is in everyday life. This can be supported by the inability to distinguish the senses in both humans and models. On figure 18 and 19, we can see how the CoS and BERT cluster the two senses of *græsstrå*. We argue that this sense distinction is not essential to understand the meaning of the language.

Figure 18: The static embeddings of two senses of *græsstrå* (green) and the word2vec representations of their hypernyms and related words (red). The dimensions are reduced with PCA.

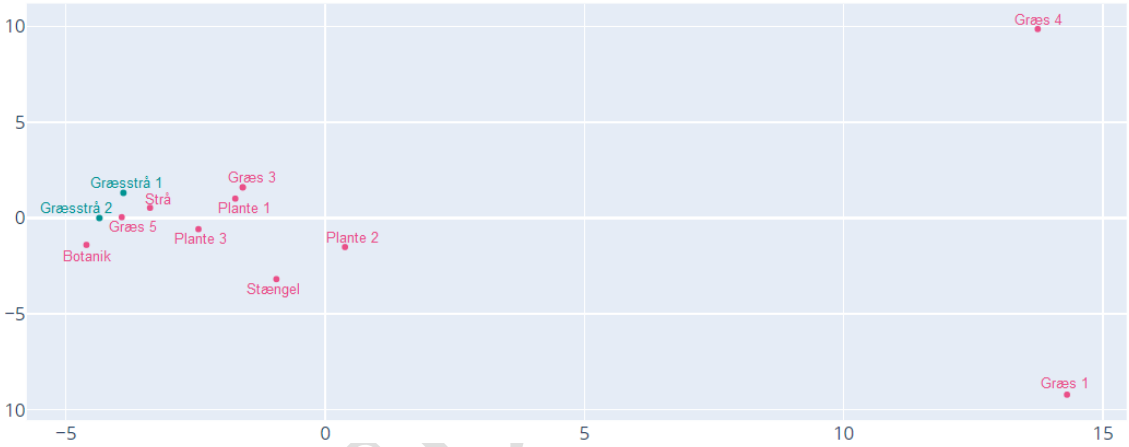


Figure 19: The contexutalised embeddings using the example sentences for representing the senses. Hypernymns and related words are also shown. The dimensions are reduced using PCA.

6.2 Extrinsic evaluation

The extrinsic evaluation tests the sense embedding methods in a WSD task, which is crucial for successful sense-tagging.

6.2.1 Experiment 4

Experiment 4 test the CoS approach on a WSD task using static and contextualised sense embeddings.

Static sense embedding

Figure 20 shows the average WSD accuracy across the 17 target lemmas for selected static embedding models. No model manages to beat the most frequent baseline (0.538) across all targets. The highest accuracy of 0.341 is achieved by a combination of local definition, local example sentence(s), and HH definitions.

This is similar to the results in experiments 1, where the *sentence-based* knowledge source (i.e., definitions) contributed the most to a higher performance. In isolation, the best knowledge source is the collection of the HH members (0.329). However, the HH members combined with the local example sentences obtains the lowest accuracy of all combinations.

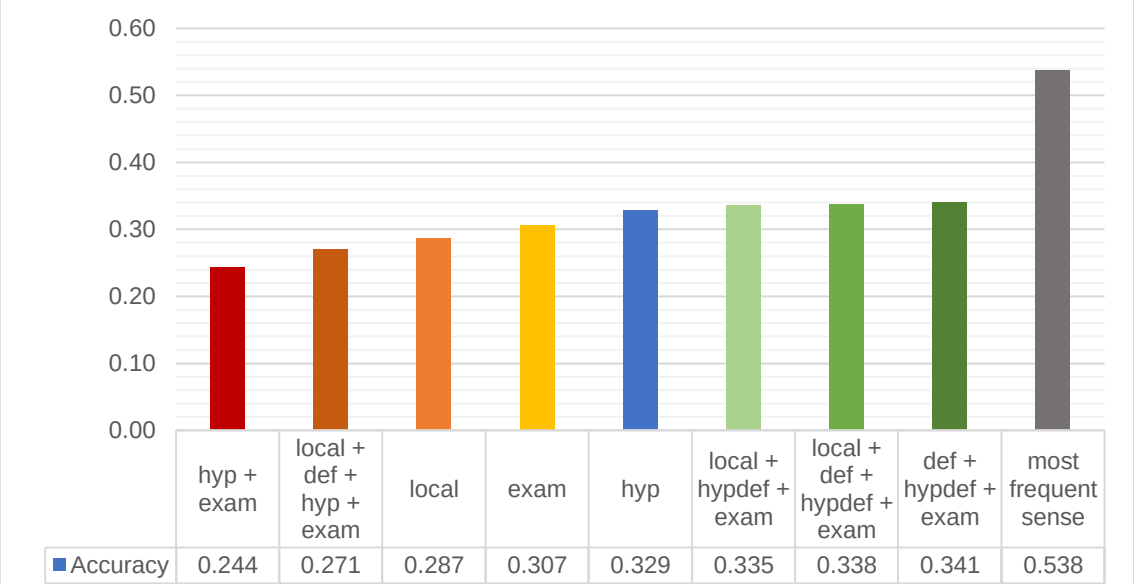


Figure 20: Average WSD accuracy for selected versions of the static embeddings across the 17 target lemmas. The figure shows the results for the representations that use a single knowledge source (local, exam, hyp), the two worst and three best performing versions, and the most frequent sense baseline.

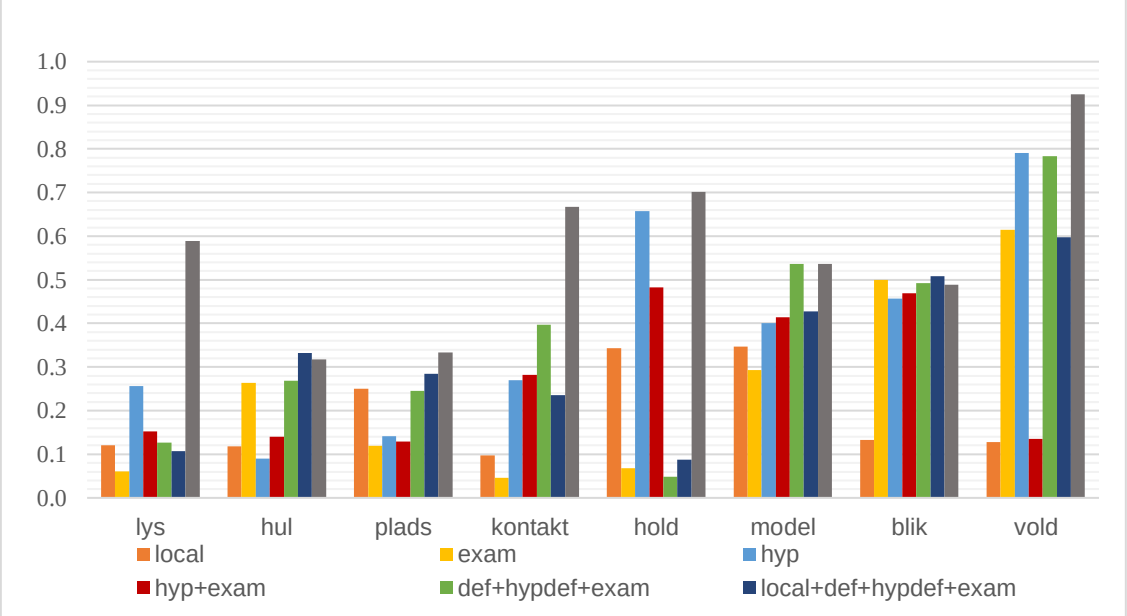


Figure 21: Accuracies on selected target lemmas

To investigate the negative interaction between the knowledge sources, we plot the accuracy on 8 of the target lemmas for a selection of static sense embedding models

in figure 21. Here, we can see a large difference between the accuracies of the example sentences (yellow) and HH members (light blue) alone. One knowledge source is not consistently better than another across the 8 lemmas. For example, the HH members perform better on *lys*, while the example sentences perform better on *hul*. This explains the low accuracy of the combination of example sentences and HH members, since the combined representation will be negatively influenced by one of the knowledge sources. Additionally, both of the knowledge sources may obtain a decent performance individually, but still result in a low performance combined (see *vold*). This indicates that these knowledge sources contain conflicting information. Each knowledge source may be able to create a coherent semantic space individually. These semantic spaces will then merge when multiple knowledge sources are combined. If the two original semantic spaces place the same sense in different positions, the previously clear distinctions of one space may be washed out by the other.

Looking at baseline, the best model (green) outperforms the baseline in *model* and *blik*, while the next best model (dark blue) beats the baseline in *hul* and *blik*. In these models, we do not see a negative effect of combining example sentences with the HH definitions. The definitions in these combinations seem to add relevant information for disambiguation for some target lemma (*kontakt*, *model*). However, this still appears to be dependent on characteristics for the individual target as visible from the accuracies of *hold*.

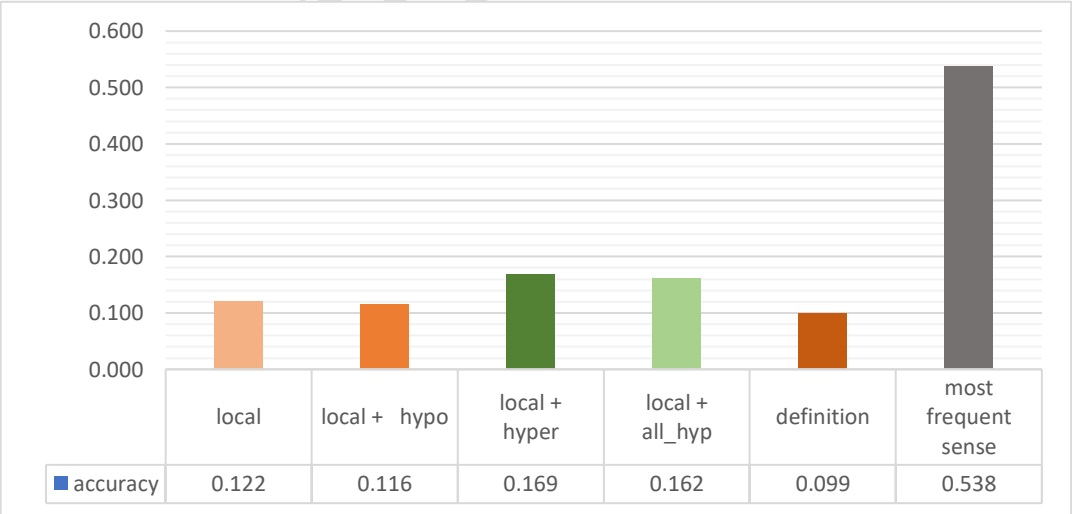


Figure 22: Average WSD accuracy for contextualised representation in the Cosine Similarity setup.

Contextualised sense representations

Figure 22 shows the average WSD accuracy across all target lemmas for the contextualised representations. The best combination uses hypernym example sentences (0.17). However, none of the contextualised representations outperforms neither the most frequent sense baseline nor any of the static representations from figure 14. The contextualised sense embeddings are therefore not useful for WSD in the CoS setup.

6.2.2 Experiment 5

Experiment 5 investigates the WSD in a NN setup with a sense-selection objective. Each training instance is a context sentence paired with every potential sense. First, the results for all models are presented. Hereafter, we take a deeper look into the training the BERT model on different data.

All models

The average accuracy for the three models across a 5-fold cross validation can be seen in table 13. The best performing model is the fined-tuned BERT with an accuracy of 0.81. This is higher than the most frequent sense baseline and all the models from experiment 4.

	Training set	**Test set**
Word2vec NN (DanNet senses)	0.22	0.22
Word2vec NN (DDO senses)	0.41	0.40
BERT FT (DDO senses)	0.92	0.81
Most frequent sense (DDO senses)		0.40

Table 13: Accuracies on the SemDaX dataset using 5-fold Stratified Cross-correlation.

None of the word2vec NN models are able to converge, and therefore achieve low accuracies for both the training and test sets. The DanNet-based static sense embeddings appear to be the most challenging for the setup. All the sense embedding models from experiment 4 achieve better than the 0.22 accuracy we report here[32].

[32] As stated under section 5.3.5, we use the example sentences from DanNet alone.

Deeper look into BERT

Since we can extract both the example sentences and definitions from DDO, we can investigate whether one of them or a combination of both is better for modelling senses for sense tagging. The combination of both gives the advantage of getting more training instances from the same resource. Table 14 shows results using the same BERT FT setup with different training and testing data. Example sentences and definitions achieve a similar performance[33], however the extra data from the combination of the two achieves the highest accuracy of 0.83.

To test the generalization abilities of the models, we create two additional tests. First, we test the models on unseen data by splitting the target lemmas into two groups. The performance of the models decreases rapidly on this test. However, the accuracies are still better than the word2vec NN models and the cosine similarity approach using static embeddings from experiment 4.

The second test evaluates how the models trained on example sentences and definitions perform on the opposite sentence type. The models can better generalize to seen target lemmas that are trained with the opposite sentence type. The example sentence model is better with a 0.67 accuracy compared to 0.61. However, there are still space for improvement.

	Examples	**Definition**	**Combined**
Cross validation	0.81	0.81	0.83
50% unseen	0.44	0.45	0.42
Trained on definition	0.67		
Trained on examples		0.61	

Table 14: Accuracies on the SemDaX with different training-test splits.

[33] Example sentences performed marginally better.

7 Discussion

In this section we will discuss the overall results of the experiments in connection with the research questions. Then we will discuss the limitations of the models, the problems with the data and suggestions for future work.

7.1 General findings

We will start by discussing the general findings in relation to the research questions.

7.1.1 Making meaningful sense representations.

The first research question aims at investigating whether we can make meaningful sense representations, by combining distributional embedding models with lexical resources. The results show that both the best CoS model (static) and the BERT FT model (contextualised) perform similarly with accuracies around 0.6. Their performance is still relatively poor when compared to the human participants (10% gap to close).

We hypothesise that the contextualised BERT model is better at creating meaningful sense representations than the static word2vec model. The BERT model has a complex architecture that can contextualise the tokens, because of the dependencies it can capture with the self-attention mechanism. However, the results from the intrinsic evaluation show that the BERT FT model is not able to outperform the best CoS model. The fine-tuning can fit the model to the training data; however, this does not translate to good performance on the test set. This is unsurprising, as it is optimistic that 1020 training instances would be enough for the model to generalise to the 1501 unique lemmas in test set. The lack of coverage to the data very likely makes the model overfit to the training data. Experiment 5 shows that a BERT model can successfully be trained to not only distinguish, but also to label, senses in a highly polysemous setup. Therefore, we expect the relatively poor performance of the BERT FT model to be due to the poor training data.

Another hypothesis is that the inclusion of DanNet knowledge can make sense representations more distinct. The CoS model has a bias towards the 'same sense' label, which shows that the model has trouble distinguishing senses. This can partially be attributed to the fine sense distinctions in DanNet and partially to the embedding model. The word2vec model still conflates senses into the same representation.

Though, we attempt to separate the senses in our sense embeddings, the semantic space is still compromised. For instance, if a sense-bag *sb* is a multiset of the lemmas $\{l_1, l_2 \ldots, l_i\}$, and one of these lemmas l_1 contains another sense that is not linked to the synset, then the final representation of *sb* will not be a one-to-one representation of the synset. To solve this problem, we need to disambiguate the senses of the lemma in the word2vec representation before representing the synset, which would in turn make the sense representation redundant.

When we examine which knowledge sources from DanNet are the most useful for distinguishing senses, we find that the 'sentence-like' sources works the best in combination. With 'sentence-like', we refer to the definitions (local + HH) and example sentences, as these consist of sequences of language, in contrast to the individual synset members. In the extrinsic evaluation (experiment 4), the results even show that the combination of all three sentence-like sources without the local synset members are better at disambiguating senses. This implies that the Qualia structure in the DanNet synset is not the most important for determining the senses computationally. However, it might also just be the case that some synsets do not contain enough lexical information to make good sense representations. We have seen cases of relative few local synset members (down to a single), and a high number of HH members (300+). Not all the HH members are equally relevant for representing a target sense, and the irrelevant members can introduce noise to the sense representation. A possible solution is to include a weighting of the members based on a score of relevance.

7.1.2 Best model for sense-tagging

The second research question examines whether the static or contextualised embedding model work the best for sense tagging, when combined with a lexical resource. We do this by testing the disambiguation abilities of the models in a preliminary WSD task on a subset of 17-18 highly polysemous nouns. We hypothesise that the BERT model would perform the best with a sense-selection objective. The results showed that the BERT model indeed performs the best with an 0.81 accuracy when trained on example sentences. Training on both the example sentences and definitions further improves the result (0.83). To our knowledge, this is the highest performance achieved on the SemDaX lexical sample subset with the full DDO sense inventory. However, the model still struggles to generalise to unseen lemmas. This

implies that the model must have evidence of the specific senses of a lemma to disambiguate it and cannot be used on unseen lemmas.

The static CoS models did perform in the expected range set by earlier research (Olsen, Pedersen, & Sayeed, 2020). We expect the poor results to be a combination of a suboptimal sense inventory for the task (i.e., unable to handle fixed phrases), the conflated senses of the word embedding model (word2vec), and the insufficient knowledge from DanNet. The CoS models are not able to efficiently disambiguate senses and would probably not be useful for the sense tagging of highly polysemous nouns.

Additionally, the results show that the static sense embeddings infused with DanNet cannot be successfully integrated in a NN pipeline (DanNet based word2vec NN). The model is not learning during training, which accounts for the poor performance. This also occurs with the DDO based word2vec NN. The failed training can either be caused by the model architecture, training parameters, or the lack of useful information in the sense embeddings. This has not been further exanimated as we do not think the approach can be much improved. Furthermore, as these sense embeddings are infused with DanNet knowledge, this setup also uses the DanNet sense inventory with a mapping to SemDaX senses. This can yet be another reason to the lower performance of the model.

7.2 Sense inventories

A major factor in all the experiments are the sense inventories. All experiments are to some degree based on lexical resources as sense inventories, the DanWiC task in terms of whether two contexts are considered examples of the same sense, and SemDaX in terms of the annotated sense labels. Additionally, we use the same resources to create the sense representations in the models. These lexical resources are rich in information and overall are of high quality as they are handcrafted, though they have limitations.

First, they have fixed, finite, and discrete sense representations. This causes the problems stated in section 2.1.3, and the systems are restricted to only handle the specific set of senses in the inventory. For instance, the CoS models can only represent senses that are present in the lexical resource (DanNet). To handle external senses, we must map the new sense to an existing DanNet synset. This is especially a problem when representing the fixed phrases as their closest literal sense in experi-

ment 4. Any future senses that may naturally occur in the language also need to either be added to DanNet or merged with a current synset.

The BERT models are less influenced by the fixed and discrete nature of the sense inventory. Not only are they trained on a vast amount of language data and therefore have knowledge about language use, but they also directly model the senses after the specific context. In a case of a new usage pattern, they may be able to infer a new sense. However, they are still influenced by the senses present in the pre-training and fine-tuning data, and it is not certain whether a BERT model can generalise new usage patterns to new senses like humans.

They can still be an advantage in framing a system onto a specific sense inventory for developmental reasons. First, we can still extract relevant knowledge from the lexical resources. Though, the sense inventory is not perfect and will never catch all senses in a language, it approximates the most frequent ones if the resource is compiled using corpus methods. Secondly, the comparison and integration of different semantic computational systems requires the same or similar sense inventory. This is especially important when working toward a common goal of improving language technology for a lower resource language like Danish. With fewer resources, it is key to coordinate every effort. Currently, multiple projects in Danish lexical semantic technology have been framed on the DDO and DanNet resources (Pedersen et al., 2009; (Schneidermann, Hvingelby, & Pedersen, 2020) (Olsen, Pedersen, & Sayeed, 2020)).

Another limitation is the sense granularity of the resources. The analysis in section 6.1.3 shows that the DanNet sense inventory is too fine-grained in some areas for both human participants and the models. This further supports the finding in Pedersen et al. (2018) that a clustering of main and subsenses can improve both human agreement and the performance of computational models. However, their approach lacks a clustering approach to adjectives like *rund* and *naturalistisk*. Since the ontological type is not useful in those cases, we suggest a clustering approach after the same hypernym and a similarity between the definitions and example sentences, perhaps represented with an embedding model.

The WSD setup also shows potential in the sense-granularity discussion. Though, it does not have perfect performance, the BERT FT does manage to get above 80% accuracy on a hard WSD task with fine-grained senses. This could possibly be im-

proved with coarser sense distinctions. Additionally, the setup is adaptable to different sense granularities. The model can easily be adapted to another sense inventory like DanNet. We only tested the DanNet senses in this setup by using the sense representations, due to the problem of fixed phrases. However, it only takes pairing the contexts with DanNet synsets instead of DDO senses to change the sense inventory.

7.3 Models

7.3.1 Comparing the models

It is problematic that the word2vec (Sørensen & Nimb, 2018) and BERT model (Devlin et al. 2018) are trained on different data. The word2vec is trained on a corpus that is better balanced as it was not crawled from the internet[34]. The proportion of newswire data is also closer to the data in the SemDaX dataset, which could give the word2vec model a slight advantage. It is however not enough for it to beat the BERT model in the WSD task. The training data problem suggests that we cannot directly compare the performance of the models. We argue that we would not be able to do that anyway, since one is a word-level model and the other does sequence modelling. The tasks need to adapt to such a degree that the differences in performance could be attributed to many other factors. Of course, the same training data can inform us how far we can get with the individual model.

7.3.2 Problems with the NNs

In general, the feed forward NNs did not perform well in any of the experiments. We contribute this to the architecture and the training of the model. The word2vec NNs had a major flaw, the model was not informed which dimensions belong to which sentence as the sentence representations were just concatenated without any separation token. Additionally, the only information these models had for the sense representations (except for in experiment 5) were a single example sentence for each sense. Arguably, using just one sentence introduces too much noise from that specif-

[34] DSL has one of the highest quality corpus on Danish with a broad coverage of different types of texts.

ic context. This is a less of a problem with the BERT model, as it is designed to classify in a two-sentence set up. With the addition of the special [TGT], we also steer the presentation toward the target. Though we can still not guarantee that the BERT model is not biased by the topic of the sentences.

The failure to train some NNs can be attributed to many factors. First, the model structure may not be deep or advanced enough to capture the patterns in the data that would result in a better performance. The choice of the loss function can also cause problems. In the binary classification tasks, we use the Mean Square Error Loss. This has the problem of potentially getting stuck on suboptimal local minima (Goodfellow et al., 2016).

7.5 Limitations with the WSD approach

The WSD approaches in this thesis have limitations.

First, we only used the sentence (CLS) representation of BERT for the last hidden layer. For simplicity, we did not extract the BERT representations for the special token [TGT]. These tokens may be a better representation as they are more local to the target lemma in the sentence. We can therefore expect their embedding to be a more accurate representation of the meaning of the target sense, in contrast to the CLS token that represents the meaning of the entire sentence and may capture irrelevant information for the target sense.

Secondly, the lexical sample WSD task was only tested on a small subset of Danish nouns. For that reason, the developed BERT WSD system can only be applied to those 18 nouns. Since the results showed that the model cannot generalise well to unseen words, we need to train the model on a training set with a broader coverage of nouns. Additionally, the WSD system is only tested on nouns. There is no guarantee the approach will convert well to other content word classes[35]. However, we are not aware of any other Danish dataset that is annotated with senses. This is an example

[35] Indeed, current work (as of November 2021) in the COR-S project does indicate a difference in the sense patterns of nouns, verbs, and adjectives.

of the knowledge acquisition bottleneck, where the required labelled data is expensive and hard to get (Camacho-Collados & Pilehvar, 2018).

The DanWiC dataset does have a broader coverage as it includes thousands of lemmas across three word classes (noun, verb, adjective). The labels are inferred from DanNet during compiling, which avoids the need for human annotation. However, the data only contains a handful of examples of each lemma. The strength of the sense-selection objective is that the model is informed of a range of context-sense relations at each training instance.

We propose another sense-selection objective built entirely from DanNet or another lexical resource. Like in DanWiC, we can extract example sentence pairs from DanNet. However, instead of only pairing each example sentence with one sense, we will pair each example sentence with all other example sentences from all synsets of a target lemma. This will be presented to the model as one mini-batch like with the WSD setup. We can then train a BERT on all the synsets that have example sentences. The labels will be inferred from DanNet as in DanWiC. In that way, we can get a broad coverage of senses in a dataset without the need for human annotation. The same can potentially be done with other lexical resources, given that enough example sentences or definitions are present.

8 Conclusion

The thesis sought to find an optimal approach for creating Danish sense representations from a combination of distributional word embedding models and handcrafted lexical resources. The focus was on finding the best approach for future sense-tagging of Danish. Therefore, we explored the combination of either a static word2vec or contextualised BERT model with the Danish wordnet DanNet and the Danish Dictionary.

The first goal was to investigate whether we could build meaningful sense representations by combining the embedding models with the lexical resources. The results from an intrinsic evaluation using DanWiC showed that both the static and contextualised models could be used to create meaningful representations. However, the better human performance strongly suggests that these models can be improved to achieve better performance. The BERT model especially performed worse than expected. We propose that additional training data and another setup could improve the BERT model.

From the DanNet knowledge sources that was used in the static representations, we found the example sentences combined with definitions worked the best for distinguishing senses, even outperforming a BERT. Therefore, we also expect that a coarser-grained sense inventory and a knowledge selection algorithm could improve the quality of the static representations.

The second goal was to find which of the two embedding models (static or contextualised), would work the best for sense-tagging. We examined this research question with an extrinsic lexical sample WSD task on a subset of 18 Danish nouns. The results showed that the BERT model got a high accuracy of 81% using example sentences from the Danish Dictionary. The performance improved further when we added definitions to the training set (83%). This proved that a BERT model can be adapted to a semantic task with sufficient data and setup. Therefore, this model appears to be the most promising to use for sense tagging. Though the limitations of coverage and training data still need to be solved.

8.1 Future work

For future work we propose to develop further high-quality training data for contextualised models like BERT. This would bring us one important step further to developing a broad coverage sense tagging system for Danish. It would especially be interesting to investigate if we can generate new datasets semi or fully automated from current lexical resources.

We also propose that we investigate further how the representations function in the BERT models. The problem with BERT is that it is a 'black box' method, therefore it is more difficult to understand where its problems lie.

Finally, we suggest that we work further with the static embedding models. Though they did not have a great performance in sense-tagging they might be useful to understand structures in lexical resources. The word2vec model is a simpler model than BERT that requires less training data. If one can prove that this model can work on other semantic tasks, it might be a useful alternative to a deeper model which requires a large expense to acquire data.

9 References

9.1 Technological resources

DDO = *The Danish Dictionary.* Copenhagen: Society for Danish Language and Literature. Retrieved 31. May 2021 From https://ordnet.dk/ddo

word2vec = Danish word2vec. Copenhagen: Society for Danish Language and Literature. Retrieved 16. April 2021 From https://korpus.dsl.dk/resources/details/word2vec.html

DanWiC and the code for the project is available at our github:

- **DanWiC: https://github.com/Linguistcoder/DanWIC**
- **Project: https://github.com/Linguistcoder/speciale**

9.2 Other references

Agirre, E., & Edmonds, P. (2007). *Word Sense Disambiguation.* Netherlans: Springer.

Banerjee, S., & Pedersen, T. (2003). Extended gloss overlaps as a measure of semantic relatedness. *Ijcai, 3*, 805-810.

Bevilacqua, M., & Navigli, R. (2020). Breaking through the 80% glass ceiling: Raising the state of the art in Word Sense Disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* (s. 2854-2864).

Borin, L., Forsberg, M., & Lönngren, L. (2013). SALDO: a touch of yin to WordNet's yang. *Language resources and evaluation, 47*(4), 1191-1211.

Camacho-Collados, , J., Pilehvar, M. T., & Navigli, R. (2016). Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence, 240*, 36-64.

Camacho-Collados, J., & Pilehvar, M. T. (2018). From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research, 63*, 743-788.

Chomsky, N. (1957). *Syntactic Structures.* Mouton & Co. .

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement, 20*(1), 37-46.

Cook, R. S., Kay, P., & Regier, T. (2005). The world color survey database. *Handbook of categorization in cognitive science* (s. 223-241). Handbook of categorization in cognitive science.

Copestake, A. (1992). The ACQUILEX LKB: representation issues in semi-automatic acquisition of large lexicons. In *procedings the Third Conference on Applied Natural Language Processing*, (s. 88-95).

Cruse, D. (1986). *Lexical semantics.* Cambridge University Press.

Cruse, D. (2011). *Meaning in language : an introduction to semantics and pragmatics* (3 udg.). Oxford: Oxford University Press.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science, 41*(6), 391-407.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Engberg-Pedersen, E., Boye, K., & Harder, P. (2019). *Semantik.* (1. ed.). Samfundslitteratur.

Fellbaum, C. (1998). *Wordnet: An electronic lexical database.* Cambridge, Massachusetts: MIT press.

Finkelstein, L., Gabrilovich, E., & Matias, Y. (2002). Placing Search in Context: The Concept Revisited. *ACM Transactions on Information System, 20*(1), 116-131.

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Geeraerts, D. (2001). The definitional practice of dictionaries and the Cognitive Semantic conception of polysemy. *Lexicographica, 17*, 6-21.

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research, 57*, 345-420.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press. Retrieved 17. May 2021 from http://www.deeplearningbook.org

Grice, H. P. (1969). Utterer's meaning and intentions. *The philosophical review, 78*(2), 147-177.

Harris, Z. S. (1954). Distributional structure. *Word, 10*(3), 146-162.

Hau Sørensen, N. C. (2021). *DanWiC: Compilinf of the Danish Word in Context Dataset.* Free Topic.

Huang, E. H., Socher, R., Manning, C. D., & Ng, A. (2012). Improving word representations via global context and multiple word prototypes. In *proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long)*, (s. 873-882).

Huang, L., Sun, C., Qiu, X., & Huang, X. (2019). GlossBERT: BERT for word sense disambiguation with gloss knowledge. *arXiv preprint arXiv:1908.07245*.

Ide, N., & Véronis, J. (1993). Knowledge extraction from machine-readable dictionaries: An evaluation. In *Workshop on Machine Translation and Lexicon* (s. 17-34). Berlin: Springer.

Ide, N., & Wilks, Y. (2007). Making Sense about Sense. I E. Agirre, & P. Edmonds, *Word Sense Disambiguation: Algorithms And Applications* (s. 47-73). Netherlands: Springer.

Johansson, R., & Pina, L. N. (2015). Combining relational and distributional knowledge for word sense disambiguation. In *procedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, (s. 69-78).

Jurafsky, D., & Martin, J. H. (2018). *Speech and language processing.*

Kilgarriff, A. (1997). I don't believe in word senses. *Computers and the Humanities, 31*(2), 91-113.

Kilgarriff, A. (2007). Word Senses. I E. Agirre, & P. Edmonds, *Word Sense Disambiguation: Algorithms and Applications* (s. 29-46). Netherlands: Springer.

Landis, J., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 159-174.

Lenci, A., Bel, N., Busa, F., Calzolari, N., Gola, E., Monachini, M., . . . Zampolli, A. (2000). SIMPLE: A general framework for the development of multilingual lexicons. *International Journal of Lexicography, 13*(4), 249-263.

Levine, Y., Lenz, B., Dagan, O., Ram, O., Padnos, D., Sharir, O., . . . Shoham, Y. (2019). Sensebert: Driving some sense into bert. *arXiv preprint arXiv:1908.05646*.

Lyons, J. (1979). *Semantics (Reprint.)*. Cambridge University Press.

Mallery, J. C. (1988). *Thinking about foreign policy: Finding an appropriate role for artificially intelligent computers.* Master's thesis, MIT Political Science Department.

Martínez Alonso, H., Johannsen, A., Nimb, S., Olsen, S., & Pedersen, B. (2016). An empirically grounded expansion of the supersense inventory. In *Proceedings of Global Wordnet Conference 2016.*

McCarthy, D., Apidianaki, M., & Erk, K. (2016). Word sense clustering and clusterability. *Computational Linguistics, 42*(2), 245-275.

Melamud, O., Dagan, I., & Goldberger, J. (2016). context2vec: Learning generic context embedding with bidirectional lstm. *In Proceedings of the 20th SIGNLL conference on computational natural language learning*, (s. 51-61).

Mikolov, T., Sutskever, I., Chen, K., & Corrado, G. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems, 26*, 3111-3119.

Montague, R. (1970). Universal grammar. *Theoria (Lund, Sweden), 36*(3), 373–398.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM computing surveys (CSUR), 41*(2), 1-69.

Neelakantan, A., Shankar, J., Passo, A., & McCall, A. (2015). Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*.

O'Grady, W., & Katamba, F. (2011). Semantics and Pragmatics: The Analysis of Meaning. I W. O'Grady, J. Archibald, & F. Katamba, *Contemporary Linguistics: An Introduction* (2nd udg., s. 197-237). Pearson Education Limited.

Olsen, I. R., Pedersen, B. S., & Sayeed, A. (2020). Building Sense Representations in Danish by Combining Word Embeddings with Lexical Resources. In *procedings of the 2020 Globalex Workshop on Linked Lexicography*, (s. 45-52).

Panigrahi, A., Simhadri, H. V., & Bhattacharyya, C. (2019). Word2Sense: sparse interpretable word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (s. 5692-5705).

Pedersen, B. S. (2010). Semantiske sprogressourcer-mellem sprogteknologi og leksikografi. *LexicoNordica, 17*.

Pedersen, B. S., Agirrezabal, M., Nimb, S., Olsen, S., & Rørmann, I. (2018a). Towards a principled approach to sense clustering – a case study of wordnet and dictionary senses in Danish. *In proceedings of the 9th Global WordNet Conference (GWC 2018)*, (s. 183).

Pedersen, B. S., Braasch, A., Johansen, A., Alonso, H. M., Nimb, S., Olsen, S., . . . Sørensen, N. H. (2016). The SemDaX Corpus—Sense Annotations with Scalable Sense Inventories. *In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, (s. 842-847).

Pedersen, B. S., Nimb, S., & Trap-Jensen, L. (2008). DanNet: udvikling og anvendelse af det danske wordnet. *Nordiske Studier i leksikografi, 9*.

Pedersen, B. S., Nimb, S., Asmussen, J., Sørensen, N., Trap-Jensen, L., & Lorentzen, H. (2009). DanNet: The Challenge of Compiling a Wordnet for Danish by Reusing a Monolingual Dictionary. *Language Resources and Evaluation, 43*(3), 269-299.

Pedersen, B. S., Nimb, S., Søgaard , A., Hartmann, M., & Olsen, S. (2018b). A danish framenet lexicon and an annotated corpus used for training and evaluating a semantic frame classifier. *In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).*

Peters, M. E., Peters, M. E., Neumann, M, M., Iyyer, M., Gardner, M., Clark, C., . . . Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pilehvar, M. T., & Camacho-Collados, J. (2018). WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*.

Pilehvar, M. T., & Camacho-Collados, J. (2018). WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*.

Pustejovsky, J. (1995). Linguistic constraints on type coercion. I P. Saint-Dizier, & E. Viegas, *Computational Lexical Semantics* (s. 71-97). Cambridge University Press.

Pustejovsky, J. (1998). *The generative lexicon.* MIT press.

Pustejovsky, J., & Jezek, E. (2016). Integrating Generative Lexicon and Lexical Semantic Resources. *LREC 2016 Tutorial*.

Quillian, M. R. (1967). Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral science, 12*(5), 410-430.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533-536.

Schneidermann, N., Hvingelby, R., & Pedersen, B. S. (2020). Towards a gold standard for evaluating Danish word embeddings. *In proceedings of the 12th Language Resources and Evaluation Conference*, (s. 4754-4763).

Schütze, H. (1998). Automatic word sense discrimination. *Computational linguistics, 24*(1), 97-123.

Sørensen, N., & Nimb, S. (2018). Word2dict–lemma selection and dictionary editing assisted by word embeddings. *In Proceedings of the 18th EURALEX International Congres: Lexocography in Global Contexts*, (s. 819-827).

Tang, G., Rao, G., Yu, D., & Xun, E. (2016). Can we neglect function words in word embedding? *n Natural Language Understanding and Intelligent Applications* (s. 541-548). Springer.

Turney, P. D., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research, 37*, 141-188.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., . . . Polosukhin, I. (2017). Attention is all you need. arXiv preprint arXiv:1706.03762. *arXiv preprint arXiv:1706.03762*.

Vossen, P. J., Bloksma, L., & Rodriquez, H. (1998). The eurowordnet base concepts and top ontology.

Wilks, Y., & Stevenson, M. (1997). Sense tagging: Semantic tagging with a lexicon. *arXiv preprint cmp-lg/9705016.*

Wittgenstein, L. (1953). *Philosophical Investigation.*

Yap, B., Jie, A., & Chng, E. (2020). Adapting BERT for Word Sense Disambiguation with Gloss Selection Objective and Example Sentences. *arXiv preprint arXiv:2009.11795*.

Zipf, G. K. (1949). *Human Behaviour and the Principle of Least-Effort.* Cambridge, Massachusetts: Addison-Wesley.