UNIVERSITY OF GOTHENBURG

DEPARTMENT OF PHILOSOPHY, LINGUISTICS AND THEORY OF SCIENCE

DEALING WITH WORD AMBIGUITY IN NLP

Building appropriate sense representations for Danish sense tagging by combining word embeddings with wordnet senses

Ida Rørmann Olsen

Master's Thesis:	30 credits
Programme:	Master's Programme in Language Technology
Level:	Advanced level
Semester and year:	Spring, 2018
Supervisor:	Asad Sayeed, Bolette Sandford Pedersen
Examiner:	Richard Johansson
Keywords:	sense embeddings, wordnet, word2vec, word sense disambiguation, clustering, machine learning, supervised WSD

Abstract

This thesis describes an approach to handle word sense in natural language processing. If we want language technologies to handle word ambiguity, then machines need proper sense representations. In a case study on Danish ambiguous nouns, we examined the possibility of building an appropriate sense inventory by combining the distributional information of a word from a vector space model with knowledge-based information from a wordnet.

We tested three sense representations in a word sense disambiguation task: firstly, the centroids (average of words) of selected wordnet synset information and members, secondly the centroids of wordnet sample sentence alone, and thirdly the centroids of un-labelled sample sentences clustered around the wordnet sample sentence. Finally, we tested the features of the cluster members and evaluation data in supervised machine learning classifiers.

The sense representations in all experiments generally beat the random baseline significantly, but not the most frequent sense as default. The representations made from selected wordnet synset information and synset members proved to generally give the best result, especially for those target words with rich synset information. The machine learning classifiers outperformed the sense representations significantly on the word sense disambiguation task. The best classifiers were those trained and tested on either the clustered data or the evaluation data. We conclude that the combination of word embeddings and wordnet associated data used to build a proper sense representation seems promising. However, we suggest some improvements for future work, specifically on the extracted information from wordnet sample sentences.

Preface

I would like to thank my supervisor, Asad Sayeed, for great guidance, discussions and for taking the challenge of understanding the Danish language.

I will also thank Bolette S. Pedersen, as my local supervisor at Centre for Language Technology (CST), Copenhagen University, for supporting and co-supervising the project, guide me through Danish computational linguistics and data, and welcoming me at CST.

I thank Nicolai H. Sørensen, Society for Danish Language and Literature, for developing and sharing the word embedding model.

Thanks to Luis Nieto Piña, PhD, Språkbanken, for valuable discussions in the initial phase of deciding the approach of the thesis.

Finally, for being incredibly strong and patient under the circumstances throughout my thesis work, and for always wanting to discuss linear algebra, I want to thank my partner, Laurits.

Contents

1 Introduction

A word can have several meanings. We can choose to represent those meanings in different ways: by translations, vectors, drawings, sound, dictionary entries or by other lexical resources. We as humans and members of a language society (e.g. the English speaking world) know intuitively that the following sentences use the word '*model'* in different ways:

The ***model*** *stood up, and smiled to the camera.*

The forecast ***model*** *predicts rainy weather tomorrow.*

If we were to teach computers to capture word senses as humans do, where should we start? And how would we know whether the machine considers word senses that correspond to what we as humans find meaningful? This thesis work is a step towards answering these questions on a case study on Danish word senses.

The question of how to handle meaning in a meaningful way has challenged philosophers, linguists, and their fellow scientists for hundreds of years. One challenge is to determine what meaning fundamentally is, another is to find a meaningful way to represent meaning. Yet another challenge is the question of whether we can create a system, which can process meaning the way humans do. Though it might not be possible to teach a machine to grasp meaning as we do, we can at least try to make it seem that way. That the machine can distinguish word senses, and not only word forms. As a stepping stone for this thesis, it is assumed that there exists an ideal quantifiable sense representation, by which computers can process word senses, and thereby overcome word ambiguity. The system creates sense representations, whose quality are determined by how well they can be used to disambiguate ambiguous words. The sense representations are created by combining knowledge-based information of words from a lexical resource with the distributional information of the word found in a corpus by using deep learning.

1.1 Focus

The aim of this thesis is to determine the quality of word sense representations created from Danish corpora in combination with a lexical semantic resource. This paper examines a word sense representation system that takes a word2vec space (Mikolov 2013) and the Danish wordnet, DanNet (Fellbaum 1998, Pedersen et al. 2009) as input, and outputs sense embeddings, which are evaluated in a word sense disambiguation (WSD) task. Finally, the sense representations will be tested in a number of supervised machine learning classification tasks. The focus will be on whether a NLP-based approach to represent word sense with wordnet senses and word embeddings is appropriate for Danish sense tagging, as well as on the challenge of evaluating the quality of a word sense representations for a relatively low-resourced language, and thereby provide a pilot project for further work on semantic processing within Danish NLP. Consequently, the focus is not on optimizing the different models and algorithms, but rather on getting an idea of whether the approach of this project is desirable for future development in the field.

1.1.1 Problem statement

This thesis intends to determine the quality of a word sense representation approach, where different wordnet associated information and word embeddings are used to represent Danish word senses. These sense representations are evaluated in a WSD task on a human annotated test set. The problem statement is therefore as follows:
Is it possible to create appropriate word sense representations by combining wordnet-based information with the distributional information of a word?

Hypothesis

It is expected that wordnet associated data provides useful information for the word sense representation system, but it is not expected that the system will beat the performance of auto-tagging with the most frequently annotated sense, as that usually is a very strong baseline in terms of accuracy. As the task of WSD of some of the most ambiguous nouns in the Danish language is rather difficult, also for humans, it is not expected for the system to perform perfectly, however, significantly better than by chance. Furthermore, it is expected that the WSD works better on the words with fewer senses.

1.2 Motivation

Computational semantic analysis systems are useful for NLP since they automatically analyse meaning in natural language. Language technology can by information of meaning (to a certain extent) tackle e.g. senses of various linguistic units, similarity and meaning relations, intended meanings, metaphors, irony etc. This possibility can first and foremost solve WSD and word sense induction (WSI) tasks, that provide the possibility of developing sense-taggers. Such a tagger can, besides getting access to sense distribution statistics, also improve other downstream applications like automatic translation, information retrieval, question-answering systems, and speech recognition. An implementation of a word sense representation system using word embeddings and DanNet is useful for further work towards that purpose within Danish NLP. This paper contributes with an idea of the quality of the sense representations created, and the thesis work can provide a starting-point for further research on WSI methods in Danish computational semantics and NLP in general.

This thesis idea was conceived during my time at Centre for Language Technology (CST), Copenhagen University, working on a project (Pedersen et al., 2018). The project was a study on how well word senses clustered by their wordnet ontological type were meaningful and useful, and whether a machine learning model could learn the features of the clusters in order to classify dictionary sense annotated sentences. This rather human-driven approach of clustering word senses, raises the question of how Danish word senses behave and cluster in raw data, without any human supervision or decisions. With the knowledge of that behaviour, a word sense distribution can be found (and hence the most frequent sense, that shows to be a strong baseline), a comparison of the clusters made by lexicographers and clusters in raw data is possible, and development of new evaluation data can take place on that foundation. As evaluation data is a must to determine the quality of any semantic analysis system, it is not possible at this stage to evaluate a completely unsupervised WSI system for Danish with curated open-source data. However, a knowledge-based system is possible to evaluate and compare with the before mentioned work, and will therefore be the product of this thesis work. This work is the first study on building Danish word sense representations from word embeddings using wordnet associated data.

One might wonder why researchers bother to investigate representations, similarities and relations of word senses, if the most frequent sense is a high-achieving baseline. Firstly, if sense-taggers or machine translation tools were developed based on most frequent sense, the sense-tagged corpora or translations would not contain the nuances of word senses that *are* present in language. It would not truly be more informative to always tag with the most frequent sense, than to simply stay on word level. Secondly, the wide-ranging applicability of knowledge on sense-level, will be limited by the performance of the most frequent sense, e.g. in information retrieval, where the results will be less accurate if one searched for a sense of a word, that was not the frequent sense. It is of course possible to use the most frequent sense as the default, and then change some sense-tags with some algorithm if needed – but the algorithm should know the possible improvements.

1.3 Contributions

This thesis work contributes to the following:

- Pilot project and implementation of word sense representation system for WSD on Danish corpora incorporating the lexical semantic resource DanNet and a word embedding model trained on big amounts of raw Danish data
- Quality measure of the chosen method
- A key from the dictionary senses (evaluation data labels) to DanNet synset id's
- Significant step towards finding word sense frequencies

1.4 Roadmap

The overall structure of the thesis takes the form of seven chapters, including this first introduction chapter.

Chapter 2 introduces the background of computational semantics, where theories on meaning representation, lexical semantics, word sense detection and representation, and WSD is introduced. Related work on supervised machine learning classifiers for WSD are also presented. In the second half of chapter 2, the theoretical dimensions of the computations and algorithms are laid out, more specifically the theory behind word embeddings, evaluation methods and the chosen machine learning algorithms.

Chapter 3 presents the applied material and software packages.

Chapter 4 is concerned with the methodology used for this study. This chapter is structured in two parts: One regarding the first three experiments, and the evaluation thereof, and one regarding the fourth experiment, which, in its nature, is significantly different from the former experiments.

Chapter 5 presents the findings of the experiments, focusing on the performance of the WSD which is used to evaluate the word sense representation system.

Chapter 6 is an analysis and discussion of the results presented in the previous chapter. Alongside, a discussion of advantages and downsides of the method, possible improvements and alternative methods are discussed.

Chapter 7 concludes on the thesis work, as well as suggesting further work on this research and the field.

1.5 Terminology

KL-divergence – Kullback-Leibler divergence

NLP – natural language processing

Sense vector is used interchangeably with *sense embedding* and *sense representation*. It refers to the vectors produced by the WSI system within the word embedded space.

Vector space model is used interchangeably with *word embedding model.* It refers to the model of word senses created on the basis of raw Danish text data with the word2vec software package (Mikolov, Chen, Corrado, & Dean, 2013).

Word vector is used interchangeably with *word embedding* and *word representation*. It refers to the vector in the word2vec model that goes together with the word form of interest.

WSD - word sense disambiguation

WSI - word sense induction

wordnet will be used to refer to any of the semantic lexical resources where the word senses are interlinked in a web formation (Fellbaum, 1998), such as the specific Danish wordnet *DanNet* (Pedersen et al., 2009).

2 Background

This chapter introduces the theoretical background of computational semantics, related work on word sense detection and representation, and WSD, followed by the relevant computational theory of the algorithms applied in this thesis work.

2.1 Computational Semantics

Semantics is the study of meaning in language. Computational semantics is therefore the study of meaning in language through computations, and is a sub-field of NLP, where computer science meets semantics, most often formal semantics. How to represent meaning is one of the core challenges in computational semantics, and the heart of this present thesis work: exploring word sense representations. The following section briefly introduces lexical semantics, which is often utilized in computational semantic systems, followed by a section with more on meaning representation. Afterwards, distributional semantics and vector space models are introduced to prepare for the sections on computational background.

2.1.1 Lexical Semantics

Lexical semantics is the study of meaning of words. The classic lexical semantics are concerned with topics on lexical ambiguity and semantic relations like synonymy, hyponymy, meronymy, polysemy, homonymy, etc. (Cruse, 1986; Adam Kilgarriff, 1997), which are those subjects this thesis work adresses.

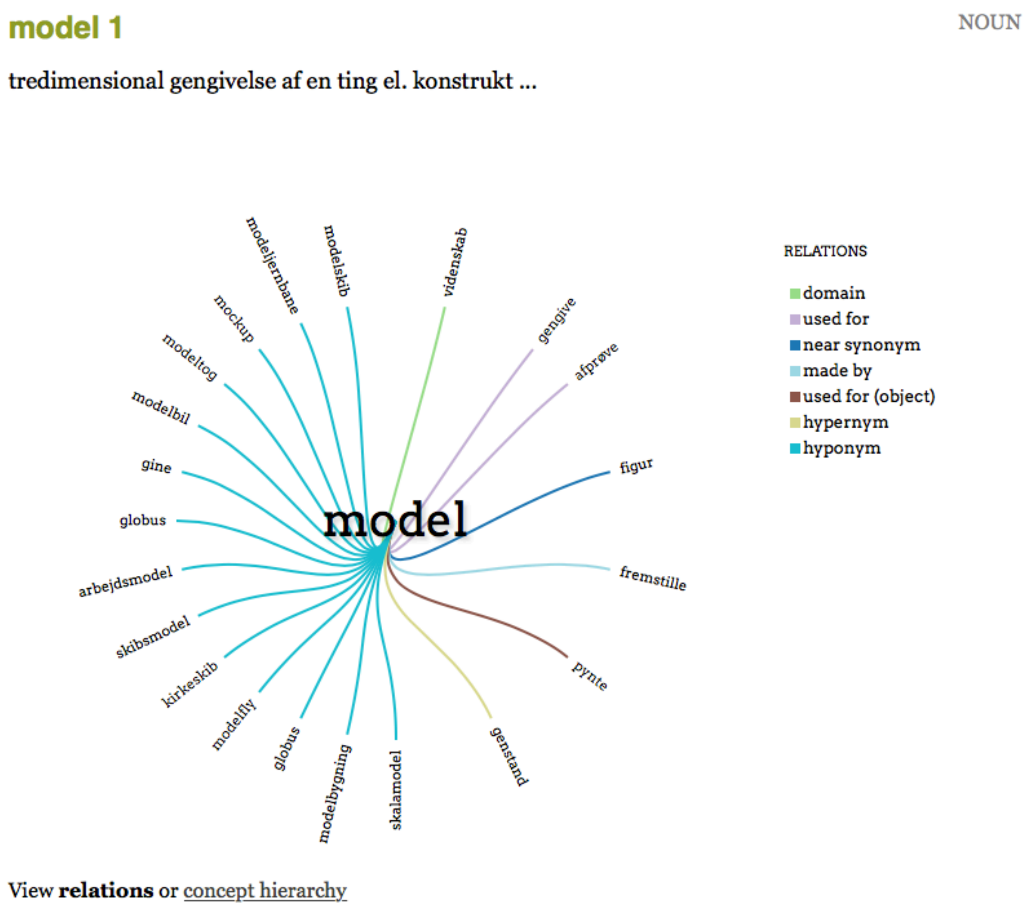
Two words are synonymous, if they mean nearly or exactly the same. Hyponymy is a hierarchical semantic relation between a generic term (hypernym) and a particular instance of that (hyponym). Meronymy refers to the semantic relation of something being a part of a whole: a meronym is something that is a part of something else. Differently, polysemy refers to the relation between a word form, and the various, but related, senses it can have. Closely related is the relation of homonymy: a set of homonyms share word form, but have different, not related meanings. The etymology of the words can reveal whether we are dealing with polysemy or homonymy. All these semantic relations between word senses is contained in the lexical database WordNet, which will be introduced shortly.

Lexical semantics is tied together with word meaning in dictionaries, wordnets (Fellbaum, 1998), and framenets (Baker, Fillmore, & Lowe, 1998) as their resources and area of research. Computational lexical semantics aims at finding the semantic relations and behaviour with NLP techniques, and use that knowledge to improve e.g. NLP applications like question-answering systems, automatic translations, etc. As high-quality computer software and open-source large corpora has become more available, the possibility to perform computational lexical semantics with less supervised methods is greater. This thesis work is a part of computational lexical semantics, as it aims to benefit from lexical resources to represent word meaning in an automatic and computable, yet meaningful way. The resource is the Danish wordnet, DanNet (Pedersen et al., 2009).

WordNet: A Lexical Semantic Resource

WordNet (Fellbaum, 1998) is a lexical semantic resource consisting of a network of so-called synsets. The synsets represent concepts, and are interlinked with several types of semantic relations (synonyms, hypernyms, hyponyms, etc.). This entails, that if a word is polysemous, then the word form is a member of several synsets. As opposed to a dictionary, where senses are structured into main and sub-senses, WordNet is rather unstructured, and has a flat structure by treating each synset equally.

Here is a visualization of a synset of the word '*model'* (as in a prototype model), where the semantic relations and a definition are shown.


model 1

NOUN

tredimensional gengivelse af en ting el. konstrukt ...

View **relations** or concept hierarchy

Figure 1: 'model' in DanNet in WordTies in its 'prototype/construction' sense and with semantic relations.

Wordnets are a valuable resource for semantics in NLP, since it provides a web of the semantic relations and structures in a formal language across word meanings in a language. The WordNet source is used across many languages, e.g. the Danish WordNet, DanNet. See section 3.3 for details on DanNet. DanNet was compiled semi-automatically from the Danish dictionary *Den Danske Ordbog* (E. Hjorth et al. 2005), but is slightly more coarse-grained than the dictionary.

2.1.2 Meaning Representation

Various systems to and theories of how to formally represent meaning have been proposed. In the following section, theories of representing linguistic meaning is briefly given. Firstly, the logic-based approaches, secondly by distributional models and formal lexical semantics.

An example from traditional formal semantics is Montague semantics (Montague, 1970). This approach and position states that there is no theoretical difference between natural languages and formal languages (like formal logic and programing languages), and they can be treated the same way. Formal logic is the study of inference, where the structure, relations and form of an expression is analysed in a strict mathematical way to determine its validity (Carnap, 1947; Frege, 1892; Kripke, 1980; Wittgenstein, 1921) Regardless of what the different entities (variables) in the expression refer to, their relative roles and how they affect each other is studied. Expressions are formalised with quantifiers, predicates, connectives etc. The meaning of a sentence formalised this way will therefore have more to do with how the variables relate, than what the variables are, as the variables can be interchanged with some of the same kind. Consequently, and broadly speaking, NLP techniques using formal logic or lambda calculus, like *cooper storage* (Barwise & Cooper, 1981), is better at handling the semantics of function words, rather than the semantic similarity of the variables, as these are "just" variables.

Vector space models (see more in 2.1.4) of distributional semantics (see next section) is a very different and rather data-driven way to model semantics, and are widely used in semantic analysis systems in NLP. Here, linear algebra is used as a tool to geometrically model the similarity of linguistic units such as words, sentences or documents. The closer the units appear in the model, the more they co-occur in the training corpus. Performing computational semantics this way more easily allows similarity measurements of the linguistic entities compared to how logic treats meaning. In other words, it is better at handling the "content" of words. A disadvantage is, that as function words are non-significant in this design, they are harder to semantically analyse. Nevertheless, some work and discussion on this issue in NLP does exists (Tang, Rao, Yu, & Xun, 2016).

An approach, which to an extent integrates both the logic-based semantics and distributional semantics, is the Combinatory Categorical Grammar (CCG) (Steedman, 2000). This grammar formalism facilitates an interface between syntactic structures and the underlying semantic representations. The semantic representations can be combined in a way that are true to the syntactic properties of a given sentence. This formalism has been implemented in various parsers, but as Clark (2014) states, it is still an open question whether logical inference or other fundamental concepts from semantics can be integrated into vector space models in a meaningful, functioning way.

Traditionally, logic-based semantics have dealt more with function words than content words, but another perspective is to distinguish between the approaches that deal with entities (e.g. lexical entities), or the combinations of it. Lexical semantics and compositional semantics (and the before mentioned distributional approach) have dealt more with the meaning content of the words, or the meaning content of the composition of the words. Generative semantics is a formal approach to access the meaning content of lexical items which claim there is a deep semantic structure by which words subsequently are arranged (Lakoff, 1971). Differently, interpretative semantics claim that meaning is derived from the set of rules that control the surface structure (syntax) (Chomsky, 1971).

So, various theoretic positions in semantics would face and represent meaning in different ways, as they focus on different aspects of word meaning: content/function words, syntax/semantics, knowledge-based/data-driven, lexical units/compositions etc. As mentioned before, lexical semantics deals with words and word senses. This could suggest that lexical semantics focus on words as entities, but this is not necessarily so. In the following section, such an example from lexical semantics is given. Here, the meaning of a word is suggested to be found through interpretation of other related relevant words. Again, this influences how a formal representation of word sense would look.

The final important theoretic framework from formal lexical semantics is Pustejovsky's theory of Qualia Structure (Pustejovsky, 1995). This interpretation of word meaning has its origin in Aristotle's theory of causality, known as the doctrine of the four causes. Here, the main idea is that a successful analysis of the world around us requires a thorough understanding of causes. The intuition is that these four factors constitute our basic understanding of an object. Pustejovsky defines the lexical semantic structure by the four interpretive levels (or formal roles), which constitute the Qualia Structure for a word:

1. **Formal**: taxonomic information. *What kind of thing is it, what is its nature?*
2. **Constitutive**: information of parts. *What is it a part of, what are its constituents?*
3. **Telic**: information of on purpose and function. *What is it for, how does it function?*
4. **Agentive**: information about origin. *How did it come into being, what brought it about?*
   (Pustejovsky, 1995)

These qualia indicate different aspects of a word's meaning, based on the relation the concept has to another word that the concept evokes. For example, the noun *child* activates conceptual relations such as having parents, being little, existing, growing, crying, playing etc. The qualia roles of *child* are those that are relevant for how *child* is used in language, which can be understood as our world-knowledge of the word. The type of this information is defined by how it impacts the word in use. According to this framework, the meaning of a word can be found by looking at the word's interpretation in context, and by exploring how these interpretations can be derived from underlying meanings when decomposing the lexical meaning into more primitive constituents (Pustejovsky & Jezek, 2016).

Pustejovsky's concept of word meaning is strongly aligned with how WordNet is built. As Figure 1 shows, the semantic relations by which concepts are linked in the semantic web (domain, used-for, made-by, used-for (object), hyper- and hyponyms), correspond to the semantic roles in the Qualia Structure. The higher you go in the concept hierarchy in a wordnet, the closer you get to the basics of the original concept, and the meaning of the concepts in a wordnet are defined by their semantic relations. A wordnet is a knowledge-based resource containing concepts which lexicographers have chosen as having a relevant semantic role for each concept. Following the Qualia Structure theory of word meaning, and if the wordnet is of good quality, it would suggest that the words in the wordnet are important for the meaning of the concepts used in language. The distributional pattern of a word, how it is used in context, might shed light on whether the hand-picked words in the wordnet have relevant semantic relations according to language use in data. This is put to a test in the method of this thesis. Not in the Generative Lexicon framework, but with the underlying approach to word meaning. See more motivation and details on this method in 4.2. Distributional semantics is described in the following section.

2.1.3 Distributional Semantics

Distributional semantics is a field in NLP that studies methods for identifying semantic similarity of linguistic units by looking at how they appear, behave and relate to each other in large corpora. Much research has been done in the field of distributional semantics since Z. S. Harris' *distributional hypothesis* (Harris, 1954), saying that words occurring in the same context tend to have similar meanings - you should know a word by the company it keeps as Firth (1957) later put it. The traditional theories of meaning considered meaning to refer to *something*, either in the external world or some mental states or intentions. The late Wittgenstein turned against this theory by stating, that meaning is its use in language (Wittgenstein, 1953). If possible at all, then meaning can be ascribed to a whole language, not to single units. This thought lies perfectly in line with the distributional hypothesis, which is a suitable theory of meaning for NLP techniques that aims at grasping meaning in an automatic way by finding patterns and relations in languages, in corpora.

Psycholinguistic studies support that human cognition treats word sense similarly as how the distributional word sense is considered in distributional semantics. It is known that if you are to identify a word, it speeds the process up if you are exposed to a semantically similar word beforehand. This phenomenon, called *semantic priming*, can be shown by various tasks, such as word completion tasks (Graf & Schacter, 1985) or lexical decision tasks (Meyer & Schvaneveldt, 1971). An example of a computational system handling and representing semantic similarity in line with the notion of semantic priming, is Latent Semantic Analysis (LSA) (Landauer, Folt, & Laham, 1998). This technique assumes that words that are semantically similar appear in similar text sections: similarity is tied to co-occurrence. The wide usage of LSA, e.g. in the field of information retrieval, supports the distributional hypothesis. Another NLP technique to handle similarity is the vector space model. These models are perfectly aligned with the distributional hypothesis, and are similar to the technique used in this thesis work to model word similarities. A deeper introduction to these vector space models is found below some final words on meaning representation and sense inventories.

The meaning of a word can be represented in various ways. Reasonably, the translation-sense of a word is the translation of the word meaning into a different language. This is useful in many everyday multilingual situations, but is not anchored in any external meaning representation. Dictionaries provide high quality and typically fine-grained definitions of word senses, and enrich many NLP tasks. Yet, dictionary senses are limited by not covering all the words used in the language, and by being rather expensive and time-consuming to create and maintain. The distributional word sense is attractive for automatic approaches in NLP by defining the word sense from a given corpus. In this way, the distributional approach to word sense representation is more perceptive to current language use, sociolinguistic analysis etc., since the input directly determines the outcome. Defining word senses this way is effective, but raises the problem of measuring the quality of these meaning representations: the lesser we as humans control or make the sense definitions, the more we need to make sure that a machine do with senses what we find meaningful.

When it comes to appropriate sense inventories, Agirre & Edmonds (2006) highlights the three Cs: clarity, consistency and complete coverage. Whether a sense inventory is appropriate or not depends of course on the application, but the inventory must be precise, have distinct representations for each sense, and ability to cover the senses apparent in language in order to disambiguate appropriately. Sense granularity is a crucial consideration to make when creating sense inventories - whether too coarse or too fine, it will cause errors. For both annotators and machines.

2.1.4 Vector Space Models

Linear algebra is a preferred tool in distributional semantics, as the linguistic units can be represented relative to each other as vectors in a geometric space, the vector space. More precisely, a vector space is a multi-dimensional space consisting of vectors, that can represent e.g. text documents, sentences, words or other instances. For example, a word vector in a vector space model, would represent a word as a point in a continuous space. The dimensions stands for a context item and the coordinates of the word represent the context counts of the word (Erk, 2012). This means, that word vectors close by each other in the space, have similar contexts – and according to the before mentioned *distributional hypothesis*, therefore also carry similar meanings.

More formally, a vector space model can be defined by four elements: a set of basic elements or dimensions (words, documents, sentences, etc.), a similarity measure between the vectors (like cosine angle, dot product, Euclidean distance), a weighting function applied to the counts, and finally a transformation of the vector space (typically dimensionality reduction to make it computationally efficient) (Lowe, 2001). So, the vectors, often called embeddings, represent the linguistic items with a low-dimensional real-valued vector.[1]

Regardless of what the vectors represent in a vector space model, they can be manipulated with the tools from linear algebra making these models attractive for computational linguists concerned with similarity and distance measures. By making use of the geometric notion of distance and linear algebra in linguistics, it is possible to make a computer handle the meaning and reasoning occurring in natural language (Clark, 2014).

A downside to these models is that we cannot interpret the dimensions of the word vectors, as it is possible to do with word context vectors (one-hot encoding) and co-occurrence vectors. For this reason, we cannot directly compare the vectors in different models over time and across different text types. An advantage is that a vector can be trained for any word, and is therefore better at handling new language use, and to include most words used in a language (given they appear in data).

An early usage of vector space models was in the information retrieval system SMART (Salton, 1968). A popular model to build word embeddings today is the Word2Vec model (Mikolov et al., 2013), which is used to make the vector space model for this thesis work, and can be accessed with the Gensim (Rehurek & Sojka, 2010) software package. The Word2Vec model is a feed-forward fully connected neural network (See more on neural networks at 2.2.3). The network comes in two architectures: the continuous bag-of-words (CBOW) and Skip-gram. The CBOW predicts the current word based on its given context in the data. The context words in (the input layer) are projected into the same point (the projection layer), and the correct middle word is to be classified. The Skip-gram model is a mirror-image of CBOW, as it tries to predict context words within a certain range of a given input word. The distant words from the middle word are given less weight, as these usually are less related to the middle word. (Mikolov et al., 2013).

[1] No transformation is of course also possible, although these are not generally considered 'embeddings'.

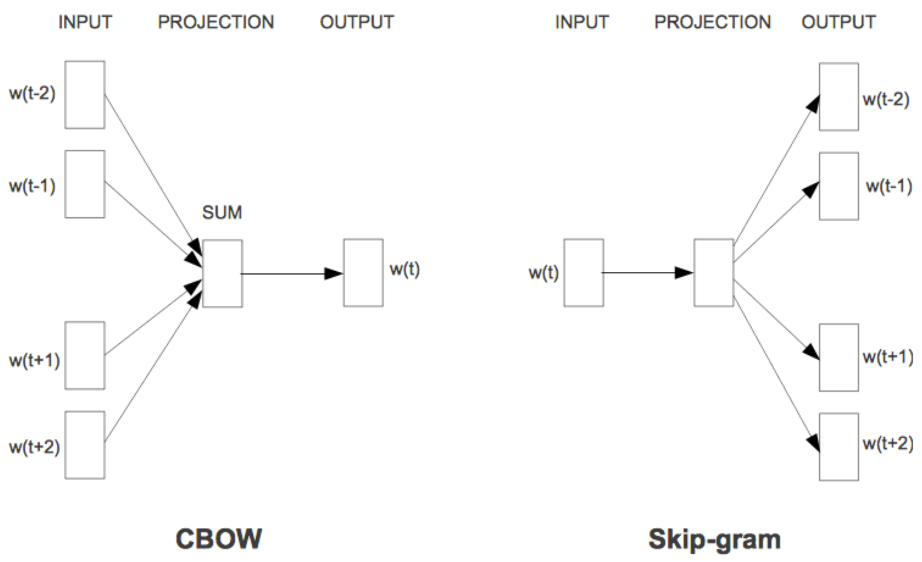


Figure 2: The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. (Picture and caption from Mikolov et al. 2013, p5)

CBOW is more computationally effective than Skip-gram, and treats all words in the window equally, but is worse at handling rare words. Skip-gram is slower, but better on less training data and at rare words and phrases. The main choices to make when training a Word2Vec model are the training algorithm (CBOW or Skip-gram), sub-sampling size (high frequency words are sub-sampled, as they often carry little information), dimensionality (of the word vectors, typically between 100 and 1000), and finally the size of the context window (5 recommended for CBOW, 10 for Skip-gram) (Google, 2013)[2].

Vector space models are usually used in information retrieval, e.g. in search engines. A simplified case: When a user types in a query, the relevant information given can be found via a vector space model where the possible documents, words or sentences are ranked by similarity to the query. Furthermore, if the query contains any polysemous words, the search engine would return more relevant information if it disambiguated the word senses at first. Each possible sense of a word can be compared to the context in which the word appears in, and hereby find the most suitable word sense. These sense representations can be induced from a vector space model (e.g. by clustering of context vectors (Schütze, 1998)). In this thesis work, sense representations are created from a model trained on a large Danish corpus, together with a lexical resource. Background on inducing and representing word sense is given in the following section.

[2] Google authors' (Mikolov et al.2013) notes on the Word2Vec project: www.code.google.com/archive/p/word2vec/

2.1.5 Obtaining Word Sense Representations

One thing is to automatically induce word senses (WSI) and their representations from data. Such algorithms group word usages according to their shared meaning. Another thing is to built them with help from lexical ressources, as in this thesis work. The task of WSI and building word sense representations is related to, but not like the task of WSD. WSI is the task of inducing word senses – to find the senses apparent in given input data of some kind. Differently, WSD is "deciding" on a sense and assign it to the given word. Sense representations, induced or built, can conveniently be tested by how well they disambiguate given words.

Several WSI and word sense representation methods have been proposed, either supervised or unsupervised, and with different input sources. Firstly, important related work on representing and inducing word sense representations is briefly introduced, and secondly the evaluation of such computational semantic analysis systems is given.

The task of automatically inducing word senses from corpora involves both to select the context features by which word similarity should be compared, and to use some technique to cluster the similar words. The word clustering technique 'clustering by committee' (Pantel & Lin, 2002) computes the top-k similar elements (word co-occurrence features) using pointwise mutual information (PMI) score, finds committees and assign the elements to the committee clusters. Another clustering technique using phrase coordination (Dorow & Widdows, 2003) where co-occurrences within phrases were considered has also been proposed. Graph-Based techniques for WSI has been proposed as well (Klapaftis & Manandhar, 2008), where edges between words is considered for clustering rather than words on their own. A different direction for WSI techniques are translation oriented (Apidianaki, 2008), where the word contexts in one language is supplemented with the equivalent features in another language. (Denkowski, 2009).

Schütze (1998) presented the first approach to automatically and unsupervised cluster context vectors (word embeddings), word sense discrimination, with the Expectation-Maximization-algorithm (EM). See description below in 2.2.1. He found centroids of clusters of dimensionality reduced word context vectors. The intuition of semantic similarity here is similar to the one in the Lesk algorithm (Lesk, 1986) for dictionary-based WSD, which choose the word sense whose dictionary definition share most words with the target word's surrounding words. The approach of this thesis work is to use wordnet associated data to create sense representations in a word embedded space, and test them in a WSD task. This approach is therefore familiar to Schütze's by grouping context vectors, but defines the word senses from a lexical resource as Lesk does. The next section will cover more recent approaches more similar to this present thesis work.

The unsupervised WSI system *SenseGram* (Pelevina et al. 2017) was recently proposed and contributes to the highly active area in computational lexical semantics that focuses on representing and deriving word meaning in an automatic way using raw data.
The technique takes a raw corpus or an existing word2vec space and automatically induces

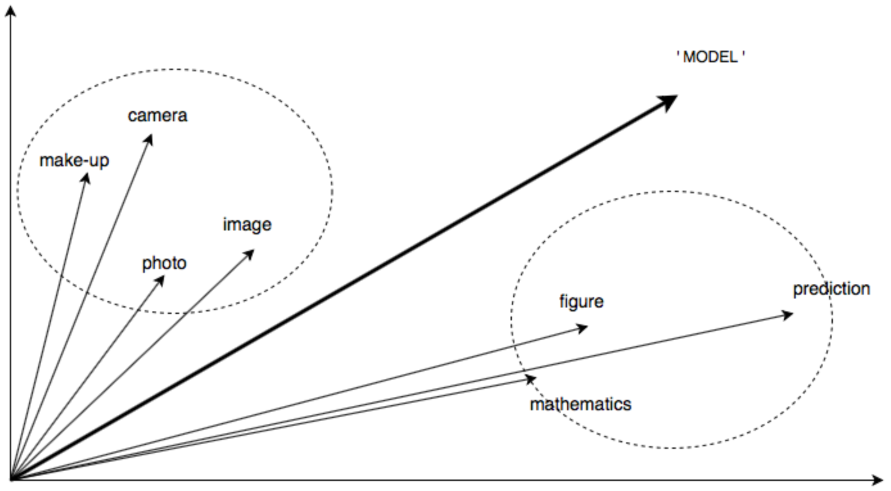


Figure 3:Visualization of the clustering technique of vector neighbours

word senses of a target word by ego-network clustering (learn word embeddings, make a graph of nearest neighbours). It performs comparably to state-of-the art WSD of the SemEval2013 data set (see more on evaluation methods below). A downside is that the system needs a set number of senses to derive. A related technique (Song, Wang, Mi, & Gildea, 2016) automatically induces sense embeddings for each polysemous word, and disambiguates a test instance by finding the nearest sense embedding in the embedded space to the instance as a contextual vector. Another related sense representation technique is the Instance-context-embedding (ICE) (Kågebäck, Johansson, Johansson, & Dubhashi, 2015) where context embeddings are created based on word embeddings and context-word embeddings computed using the Skip-gram model, and assign different weights to the context words based on how they influence the meaning of the target words of interest. This assumes that context words that tend to correlate with the target word are more important to the meaning of that word. The context embeddings are then clustered with the k-means algorithm.

Another, and significantly different, approach is as Johansson and Nieto Piña's (2015), whose system "splits" the word embeddings to sense embeddings while training with information from a Swedish semantic resource (SALDO), and keeps the found sense vectors similar to its network neighbours. It is evaluated extrinsically in a classifier for creating lexical units for FrameNet frames. This approach is similar to Bhingardive et al. (2015)'s wordnet affected system who also plug-in a resource, WordNet, to obtain word sense representations in a vector space. Note, the task in Bhingardive et al. (2015) is not WSD, but most frequent sense detection. Both approaches bootstrap the sense representations by creating them based on a lexical semantic resource, as in this thesis work. They are significantly different from the above mentioned unsupervised approaches, as they use a lexicon for obtaining word sense representations.

When using a large lexical resource to represent word senses, there is a risk of over-representing rare word senses: all the senses in the lexical resource are equally represented, even though they are not equally distributed in data (or generally used in language). Another risk is to miss corpus-specific senses. These risks make the unsupervised WSI techniques attractive. Nevertheless, lexical resources are often of good quality, and do contain information of rare words. The task at hand is therefore to identify which method to use and whether bootstrapping of the unsupervised WSI is possible.

2.1.6 Evaluation of Semantic Analysis Systems

Human handcrafted semantic data is expensive, but evaluation data of some kind is a must to determine the quality of any semantic analysis system. Computational semantic analysis systems are typically evaluated on the data sets from the ongoing series of SemEval – International Workshop on Semantic Evaluation (A Kilgarriff & Palmer, 2000). WSI systems has typically been evaluated by comparing to a Gold Standard or in a WSD task measuring the quality by performance (Agirre & Soroa, 2007). The evaluation data produced for SemEval 2013 task 13: *Word Sense Induction for Graded and Non-graded Senses*[3] is the standard data used to test WSI systems, and gives a common ground for fair comparison. The above mentioned related work is mostly tested on SemEval 2013 task 13 data.

The SemEval 2013 task 13 was to explore the possibility of perceiving multiple senses in a single contextual instance. Participating systems were asked to annotate nouns, verbs and adjectives in sentence instances using WordNet 3.1 (Fellbaum, 1998). One sense or several weighted senses could be assigned. This means that instances potentially could be labelled with multiple senses and with weights. The trial data were annotated with ratings of all senses, where each sense and instance combination were treated as a separate element to score. The systems were evaluated in two settings: (1) in a traditional WSD task (comparison of wordnet sense labels), and (2) in cluster-based evaluation (comparison of induced sense inventories to wordnet inventories)

The WSD task contained three steps. Firstly, the systems should detect the relevant applicable senses for the given instances. The Jaccard Index was applied as evaluation measure. Secondly, the detected senses should be ranked by their applicability. Here, the evaluation measure was the Kendall's τ similarity. Thirdly, the agreement between the ratings and human annotators should be measured by weighted Normalized Discounted Cumulative Gain.

In the cluster-based evaluation setting, the sense clusters induced by the systems were compared to the sense clusters annotated by humans by the Fuzzy B-cubed and Fuzzy Normalized Mutual Information measure. (Jurgens & Klapaftis, 2013).

[3] https://www.cs.york.ac.uk/semeval-2013/task13.html Retrieved 10.04.18

As the systems induce several senses, considering the aspect that one sentence might be assigned more than one correct sense, and rank them by possibility, the task is highly comparable to the work of this thesis. Furthermore, the test data produced for SemEval 13 task 13 is also annotated with WordNet senses. Different from the SemDaX data used in this thesis work, the words to disambiguate were not as ambiguous and there was higher inter-annotator agreement. Another and crucial difference is, that the senses in SemDaX are not ranked. Annotators for SemDaX were asked to assign one sense to the given instance. The inter-annotator agreement is relatively low (see Chart 1), which possibly (and partly) is due to the fact that more than one sense is applicable. To include and consider all annotations in SemDaX, all the annotations are considered correct (but unranked). The word sense representation system of this thesis work (see more details on method in chapter 4), represents all possible wordnet senses in a vector space. The representations can then be ranked by a similarity measure to a given test instance represented in the vector space, and the representation with the highest similarity score can be chosen for a disambiguation guess. The system of this thesis do not detect a group of relevant senses, but merely rank by similarity and/or pick the most similar sense. Since the test data SemDaX do not contain ranked senses, and the word sense representation system of this thesis do not choose a set (or cluster) of relevant senses, a direct comparison to the systems developed for SemEval 2013 task 13 with the same measures is not straight forward. However, it is possible to compare the n-sized set of un-ranked annotations in the test data SemDaX with the set of n-nearest sense representations in the vector space with the Jaccard Index. It is also possible with this measure to directly compare the single system guess to the (unranked) annotated senses in SemDax (though this would result in a low score for instances with many sense tags). This evaluation would be similar to the first step in the SemEval 2013 task 13 WSD task: detecting which senses are applicable.

The before mentioned *SenseGram* induced word senses given a raw corpus or a trained word2vec model, and was tested on the SemEval 2013 test data. This method is attractive by being highly applicable to other languages, but there is no open-source curated data at this stage for testing such a Danish WSI system.[4] Data for this kind of evaluation requires a set or cluster of words (or word representations) which the induced senses can be compared to. For whatever sense representation created un-supervised or semi-supervised, the senses need to be anchored to a sense label or some external criterion in order to be evaluated. Extrinsic evaluation on a specific application is also possible. The system can e.g. auto-tag new data or machine translate, and then be evaluated by users on how well the system solved the task. Internal evaluation of e.g. induced sense clusters would include scores of the quality of the clusters in themselves in

[4] A set of relevant words for each sense in an annotated corpus can of course be extracted. It would be possible to do this for the SemDaX data, though it would not be clear to what extend the relevant words within the corpus are those which are relevant globally (or in the training data for creating word embeddings). Extracting such data for testing WSI systems for Danish would be a relevant future project.

terms of class purity or silhouette coefficient, which compares the average distance of elements within a cluster to the average distance to elements in other clusters.

Baselines and ceilings

The obvious baseline to compare WSD system performance to is the frequency of the most frequent sense. Often, the first sense in a lexical semantic resource is the most frequent one – as in WordNet. It can also be found by counting the senses in a labelled corpus, as in this thesis work.[5] The most frequent sense is usually a hard baseline to beat, and is therefore often used as the default sense. Also, the before mentioned Lesk algorithm is a used baseline (Jurafsky & Martin, 2009). Finally, the random baseline where senses are chosen randomly by chance, is also used.

Human inter-annotator agreement is considered as the upper bound. At least two annotators annotate a corpus, and an agreement score is calculated. A measure, e.g. the Fleiss score or Krippendorffs alpha, is afterwards usually applied to take care of the fact that it is easier to agree on a few senses than on many. As humans tend to disagree, and we want a human created gold standard, we should not expect a better result from a machine. We need to trust the quality of the annotated data in order to rely on the models created from or tested on the data. The inter-annotator agreement is therefore a measure of this quality. This score is usually set to at least .80, but for systems disambiguating highly ambiguous words, lower agreement score is acceptable. As Poesio & Artstein (2008) writes, word sense tagging is more challenging than e.g. POS-tagging and dialogue act tagging. The same categories can be used to classify all units, but different categories must be used for each word when annotating senses: a precise coding manual specifying examples for all categories for annotators is hard to make. A help is to tag with e.g. dictionary senses, but the granularity and architecture of the sense inventories can vary across dictionaries. Again, the task at hand help identifying which inventory to use. Improvement of the inter-annotator agreement score can be reached for instance by applying a coaser grained sense inventory by collapsing dictionary entries (Bruce and Wiebe, 1998; Palmer, Dang, and Fellbaum 2007; Pedersen et al. 2018) or by letting profesional lexicographers annotate the data (Kilgarriff, 1999).

2.1.7 WSD Evaluation Metrics

The quality of WSD of test data can be measured with various statistics: accuracy, precision, recall, F-score etc. of how accurate the system guess matches the gold standard. Sometimes there are more than one correct class per instance. That is the case in this thesis work, where all annotations are considered correct when annotators disagree. In the SemEval 2013 task 13 the same idea motivated the task, namely that there can be more than one sense assigned to a target word. To include the fact that some incorrect disambiguations are better than other senses when

[5] If counting annotated senses in data, it is still an open question whether that sense distribution is generally accurate, or just the case in the specific sense annotated data.

evaluating the system quantitatively, the induced senses can be weighted. In the work of Song et al. (2016), each induced sense is compared to the test sentence centroid by Euclidean distance, where this thesis work instead uses the cosine similarity as a distance measure[6]. The similarity scores for each sense can be weighted by the distance, and utilized in the decision of which sense to choose. An example of this in use is Google Translate, that ranks possible translation alternatives when clicking on the proposed translation.

The weighted outcome of a WSI or word sense representation system can be compared to a gold standard. In the next section a distance measure between two weighted sets of elements is introduced.

Kullback-Leibler divergence

The weighted outcome of the sense representation system of this thesis is compared to the annotations that humans labelled the senses with. As the created sense representations can be weighted, they can be represented by a probability distribution. The annotations are also represented by a probability distribution, incorporating the fact that each instance can have multiple classes. A classic way to compare probability distributions through is the Kullback-Leibler divergence (KL-divergence), also called relative entropy.

The KL-divergence, D_{KL}, is given by

$$D_{KL}(q||p) = \sum_{i=1}^{N} q(x_i) \cdot log\frac{q(x_i)}{p(x_i)},$$

where q is the probability distribution and p is the approximating distribution. It measures the difference between one distribution and the other. If the KL-divergence is very low, the distributions are very similar. The measure is not symmetric, (and therefore not a distance measure, see Figure 4) and there has been some discussion on choosing p or q as the approximating distribution (Goodfellow, Bengio, & Courville, 2016).

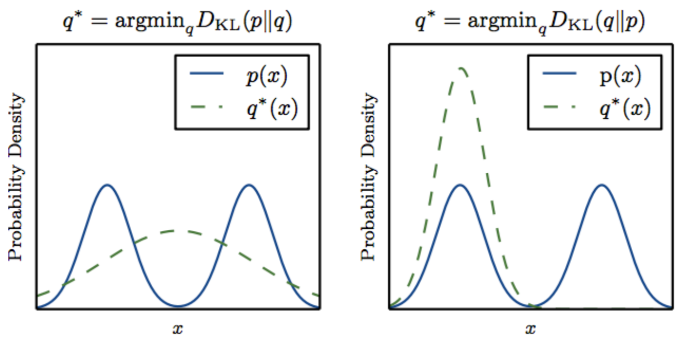


Figure 4: KL-divergence example for P||Q and for Q||P shows that the KL-divergence score is asymmetric. Figure from Goodfellow et al., (2016)

[6] The Euclidean distance could just as well be applied in this work as the cosine similarity. Intuitively speaking, the difference is that the cosine similarity cares about direction of the vectors, not considering the length of the vectors. The length influences the Euclidean distance though. When averaging vectors as in this work, the length of the vectors matters less. But if vectors are added, the choice of distance measure is more important. As always, the task at hand can help identifying the right distance measure.

Either way, the KL-divergence score is a measure of how off one distribution is to another. In this thesis setting it is how off the annotations are from the system-built senses.

Before moving on to the computational background related to this thesis, a final section on related work is found. An additional experiment using machine learning classifiers for WSD is carried out in this work. In the following section an overview of related work on dealing with word ambiguity and WSD with machine learning models is given.

2.1.8 Word Sense Disambiguation with Machine Learning

There are several approaches to WSD, which mainly fall into two groups: knowledge-based and supervised (or semi-supervised). One approach is to compare by some measure sense representations (induced from data or built from lexical resources) to the context item where the ambiguous targetword is located, as the majority of the experiments of this thesis work (see details in chapter 4). Another approach is to train machine learning models on annotated data and disambiguate new ambigous words with those models.

When inducing or building sense representations, words in multiple word items are often concatenated into one single sense representation. A consequence of this is that information of syntax and sequence is lost. To handle this problem Yuan et al. (2016) suggests an approach to WSD using neural models, namely a Long Short Term Memory (LSTM) network (Hochreiter & Schmidhuber, 1997). They presented a supervised WSD algorithm and a semi-supervised algorithm. Besides the most frequent sense as a baseline, they implemented a classifier, where they compute sense embeddings by averaging the context embeddings (produced with word2vec) of the sentence instances with that sense label. Also as in this thesis work, they use cosine similarity to compare the sense embeddings with the context embedding. This baseline is similar to the approach of this thesis, but they created sense embeddings based on labeled sentences, not from a lexical resource.[7] Their supervised LSTM model for an all-words WSD task is trained to predict a held-out word in a sentence, given the surrounding context. Their semi-supervised WSD classifier labeled unlabeled sentences from the web based on how similar they are to labeled sentences. The models outperformed the baselines significantly.

Kågebäck & Salomonsson (2016) also presented a sequence modelling approach to WSD using a neural LSTM model, though a bidirectional one. That means the classifier gets and stores information both from the left (past) and right (future) when predicting a sense for a word. Each word in the text was represented by a word embedding (not concatenated) to not miss out on sequential and syntactic information, as well as avoiding to depend on handcrafted features or external resources. The model computes a probability distribution of the possible senses of the word given in a given document. The model is trained with a limited number of word sense

[7] A future project to this thesis work could be to test this approach to represent sense embeddings: to take the average of sense-labeled instance vectors.

labeled instances, and is evaluated on lexical sample WSD tasks of the SemEval 2 (Kilgarriff, 2001) and 3 (Mihalcea et al. 2004). Differently, Yuan et al. (2016)'s model can generally be used for any word, and can therefore better achieve high performance on all-words WSD tasks.

Raganato et al. (2017) approached WSD with a different perspective. They did not view the task of WSD as a classification problem, as Yuan et al. (2016) and Kågebäck & Salomonsson (2016), but trained models with sequence learning. In this way, there is not one model trained for every target word, but one single model trained at once on the sense annotated input text. The target words are then disambiguated jointly. They developed various neural model architectures of bidirectional LSTM taggers and Sequence-to-Sequence models. The models were compared to the best knowledge-based (version of Lesk, Basile et al. (2014), UKB, Agirre et al. (2014) and Babelfy (Moro et al. (2014)) and supervised systems (*Context2Vec*, Melamud et al. (2016); *It Makes Sense*, Zhong & Ng (2010); Iacobacci et al.(2016)) tested on the same framework. Though Raganato et al. (2017)'s approach did not rely on so-called *word-expert* classifiers (models trained for single words), the performance achived state-of-the-art results.

The above-mentioned method of Zhong & Ng (2010), as well as Shen et al. (2013), are traditional supervised WSD approaches where local features around a target word are extracted and used for learning in a classifier. These are more similar to the classifiers trained for a lexical sample WSD task in this thesis work, which are trained on extracted surrounding information for each targetword, namely context vectors concatenated from word embeddings of the context words. The classifiers of this thesis work is therefore in the category of *word-expert* models.

2.2 Computational background

In this section, the algorithms applied in the word sense representation system of this thesis work is briefly introduced. Namely the k-means algorithm for clustering sentences, the machine learning algorithms used for classifiers trained on the clustered instances and evaluation data.

2.2.1 The K-means algorithm

The EM algorithm is the skeleton in the k-means algorithm (Forgy, 1982; Hartigan & Wong, 1979; Lloyd, 1982; MacQueen, 1967), which is used to cluster embeddings in this thesis work. The EM-algorithm (Dempster, Laird, & Rubin, 1977) is an iterative method to find parameters to distinguish groups in data consisting of un-known variables. The algorithm consists of two steps: The E-step where data points are assigned to generated hidden values (expected parameters), and the M-step where the likelihood is maximised to fit the observed data points.

The k-means algorithm is a classic unsupervised clustering algorithm. It is unsupervised as the data points do not have a pre-assigned class. The algorithm requires a set number, *k*, of clusters, *C*, to group the given data by. The algorithm seeks to separate groups of equal variances by minimizing the inertia:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_j - \mu_i||^2)$$

where *x* are the data points, μ the mean of the samples. In other words, it is the sum of squared distances to the cluster centroid. The clusters are described by the mean of their member data points, called the centroid. The algorithm alternates between assigning data points to clusters, and updating the placement of the centroid. The initial centroids are random (or given seeds as in this thesis work), and the algorithm iterates till the same data points are clustered in the same cluster regardless of the initial centroids.

2.2.2 Support Vector Machine

The support vector machine (SVM) (Vapnik & Lerner, 1963) is a classical machine learning algorithm for classification tasks. The model is a supervised learning model which seeks to find a perfect hyperplane in a vector space by which the classes of the given data points are separated. The method is to maximize the magnitude of the distance (the functional margin) between the support vectors separating the classes. This assumes that the classes are linearly separable, and that a hard margin can be found. But if some data points are mingling with the class on the other side of the current hyperplane, a soft-margin is helpful. A hinge-loss function adds tolerance to the hard-margin, and the values of this function considers the distance from the mingling points to the margin. With the kernel-trick introduced (Boser, Guyon, & Vapnik, 1992) the SVM can also handle non-linearly separable classes.

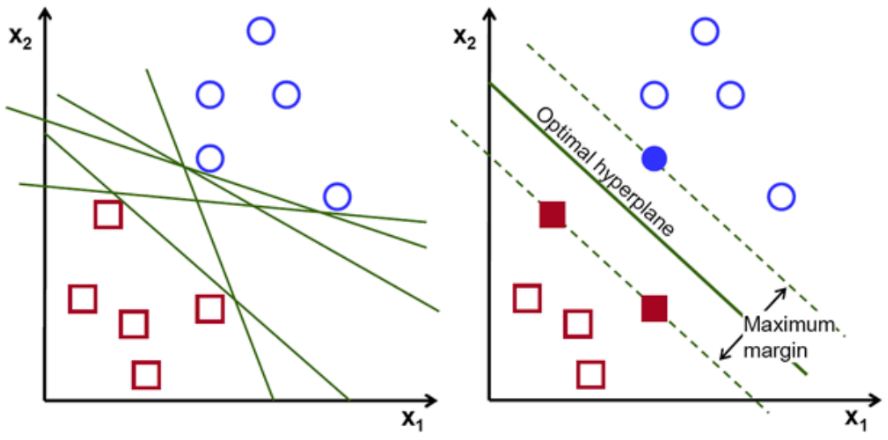


Figure 5: SVM. The green lines in the left system are possible hyperplanes, and the green line in the right system is the optimal hyperplane as this has maximized the margin of the support vectors. Figure from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 22.09.18

2.2.3 Feed-Forward Neural Network

Neural networks are increasingly popular supervised machine learning algorithms capable of modelling non-linear patterns in data. Though neural networks have been known for decades (Farley & Clark, 1954; Rochester et al., 1956; Rosenblatt, 1958), lately, deep architectures are widely popular due to increased available computing powers (e.g. GPU's). The network is good at recognizing patterns as it can handle much more information at the same time than what humans can handle and calculate. The model is brain-inspired as it consists of a network of nodes taking input and gives output, according to certain activation rules. The links between the nodes are given weights, which activates through the network leading to the output of what the network once learned such an input should end up as. The network needs a certain amount of training data to learn those patterns and weights in order to output the correct things when tested on evaluation data. Neural networks have proved useful for a range of tasks from bioinformatics to NLP, sound, and vision.

A feed-forward neural network is the first developed deep learning model, and only sends information one way through the network and do not allow nodes later in the network to update weights in earlier layers at the same training step, as recurrent neural networks does. Continous activation functions are used to activate the nodes. The loss function tells how well the network models the given data. The knowledge of the gradients of this loss function, gives access to the speed of the loss changes when changing the weights through the network. Backpropagation (Rumelhart et al., 1986) is a method to calculate the gradient of the loss function, and it distributes the loss (found at the output layer) back through the network layers. The weights can be updated accordingly.

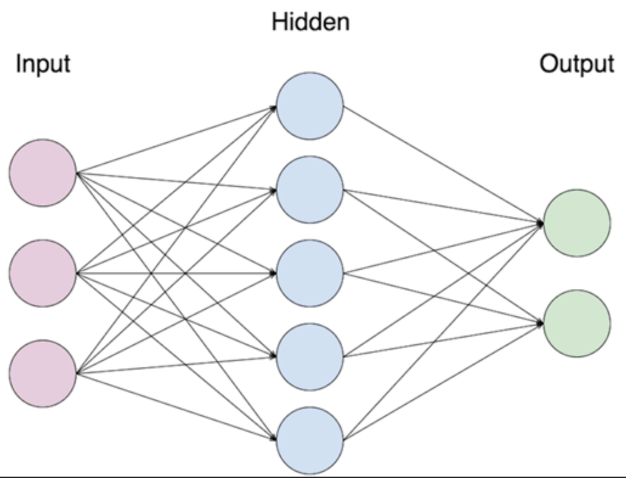


Figure 6:Neural network with one hidden layer. The circles are nodes, the arrows are the connections with given weights.Figure from https://hackernoon.com/artificial-neural-network-a843ff870338

3 Materials

We now present the materials used in this study. Firstly, the polysemous target words, then the vector space model, the Danish wordnet DanNet, the evaluation material SemDaX, the additional Danish corpus Korpus DK, and finally an overview of software packages is found.

3.1 The target words

The words of interest in this work is 17 of the most polysemous Danish nouns. That is, members of the set of nouns in the Danish dictionary having the highest number of main and sub-senses. The number of word senses varies from 8 to 30. The nouns, translations and number of senses can be seen in Table 1. By taking highly ambiguous words into account, it is possible to access how the WSD performs on a highly challenging task. Performance on the hardest task will set the bar for the quality of the approach, since the performance will increase on easier tasks.

Noun	Translation	Senses
Ansigt	Face	16
Blik*	Look, glance, tin	8
Hold	Team, side, gang	10
Hul	Hole, gap, leek	22
Kort	Card, map, plan	21
Lys	Light, candle, lamp, glare	30
Model	Model, pattern, type, design	9
Plade	Plate, sheet, disc	13
Plads	Room, space, square, post	21
Skade*	Harm, injury, damage, magpie, ray	12
Slag*	Battle, stroke, cape, roll	28
Stand	State, condition, shape, sales pitch, booth, stand	11
Stykke	Piece, part, length, paragraph	22
Top	Top, peak, apex	12
Vold	Violence, bank	10
Kontakt	Contact, switch, touch	9
Selskab	Company, party, association	11

*Table 1: Target nouns, their translation and number of senses encpuntered in SemDaX. * = homonym*

The selection of nouns corresponds to the target nouns of the available evaluation data, SemDaX (see below).[8]

[8] except for 3 words: *kurs*, *skud,* and *tang*. The manual data linking from dictionary sense labels to DanNet synsets (see section 4.1 for more details) resulted in too coarse a granularity for the nouns to be as polysemous as this thesis work aims to investigate.

3.2 Word Embeddings

The word2vec model is created by Society for Danish Language and Literature to use in another project (Sørensen & Nimb, 2018). They used Gensim and Python to train a word2vec model on a corpus of roughly 920 million running words (mostly newspaper articles, but also magazines, speeches and discussions from the Danish parliament, fiction, etc. from 1982 to 2017). The corpus had 6.3 million token types, where 5 million occurred less than 5 times.
The dimensions of the word embeddings are 500, a window size of 5, and a threshold for rare word on 5. The CBOW algorithm is applied.

3.3 DanNet

DanNet (Pedersen et al., 2009) is the Danish version of Princeton WordNet (Fellbaum, 1998). It is freely available[9], and can easily be browsed with the application *WordTies*.[10] DanNet was compiled from the Danish dictionary (Hjorth & Kristensen, 2005). At this point, DanNet consists of 66.308 concepts that are fixed by 326.566 inter-related semantic relations.
The data extracted for this thesis work is all the synsets that belong to the target nouns. For each synset, the sample sentence, the definition, and the semantic relations (hypernyms, hyponyms, domain, synonyms, near-synonyms, made-by, made-of) are extracted. The super-senses are not extracted, as those are described in English, and not in Danish words.

3.4 Evaluation data: SemDaX

The SemDaX corpus (Pedersen et al., 2016) is extracted from the 45 million words CLARIN Reference Corpus (Asmussen, 2012) and consists of different text types: blogs, chat forums, newspaper, magazines, discussions and speeches from the Danish parliament. The size of the semantically annotated corpus is 90.000 words, where newspapers make up the major part (48%).

The exact SemDaX data extracted for this thesis work is the 6012 sentences containing the target nouns, which are annotated with dictionary senses by 2-6 annotators, who are advanced students and researchers. There are 355 sentences per target noun on average, where the more senses the noun has, the more sentences are extracted. A window of 5 context words is considered, stopwords are removed, but no text normalization.

[9] https://cst.ku.dk/projekter/dannet/ Retrieved 11.11.18
[10] http://wordties.cst.dk/wordties-dannet/ (developed by Anders S. Johannsen & Mitchell Seaton). Retrieved 20.06.18

3.5 Korpus DK

Korpus DK (Society for Danish Language and Literature[11]) is a corpus of different text types in Danish, and has a size of 56 million words. It consists of relatively recent language and mostly every-day language use.

For each target noun in this thesis work, around 1000 sentences containing that noun is extracted. A window of 5 words (left and right), stopwords are removed, and no normalization is chosen in line with the pre-processing of other data in this project.

3.6 Software packages

- **Python** (van Rossum, 1995) is used to code the system implementation
- **Sci-kit Learn** (Pedregosa & Varoquaux, 2011) software packages is used for the clustering and machine learning tasks
- **SciPy** (Jones, Oliphant, Peterson, & others, 2001) for NumPy, Matplotlib, and Pandas is used for data handling and plotting
- **Keras** (Chollet, 2015) is used for deep learning implementation. Tensorflow (Abadi et al., 2015) is used backend.

[11] Retrieved from https://ordnet.dk/korpusdk 15.05.18

4 Methods

The aim of the thesis is, as stated in the introduction, to examine whether it is possible to create appropriate sense representations with wordnet based information and word embeddings. An appropriate sense representation is in this context one that adequately can perform WSD on evaluation data. In the beginning of this chapter minor practical challenges are mentioned, as they effect the method of this thesis work. Then, the data processing decisions are presented, before the mapping between the two datasets DanNet and SemDaX is described. Afterwards, experiment 1-3 of this work are described, followed by the evaluation method and statistics used to measure the quality of the outcome of these experiments. Finally, experiment 4 is presented.

Practical challenges

As for much other research, the method of this present thesis work is rather influenced by the data available. It would indeed be very interesting and valuable to explore how fully unsupervised and automatic induced word senses look and behave in the Danish language, but there is no available evaluation data for such sense representations for Danish at this stage. See more on evaluation methods in 2.1.6. However, the dictionary word sense-annotated data SemDaX, as described in the previous chapter, is available. In this work, SemDaX is used as evaluation data and the objective (See method details in second half of this chapter).
The Danish dictionary is not available open-source. But the source to the Danish wordnet DanNet is (to a certain extent) available for download. To make use of this lexical resource in this thesis work, a key between the dictionary labels and DanNet synsets needs to be established. DanNet was compiled on the basis of the Danish dictionary, and a key does exist – but is not available for research. For this reason, a key is created manually. See details in section 4.1.

Pre-processing

As described in the previous chapter, the evaluation data SemDaX, the corpus Korpus DK and the DanNet is used (besides the provided word2vec model).
All the text data used in this work is pre-processed equally. The relevant part of SemDaX consists of a number of sentence-instances per target noun. All the instances are word tokenized, as we are interested in the words in the sentence. The context with a window of 5 words is considered, just like the word2vec model processed the input data. The words are collected in a bag-of-words, without any punctuations or stopwords[12], to avoid less informative information, and is then represented as a vector as the mean of each word vector in the word2vec model. Korpus DK is processed in the same way. The sentences from DanNet, e.g. the sample sentence and definition, is processed in the same way, except that the context window of 5 is not considered. The reason is that the exact target word is not always in the same word form in the sample sentence, and not necessarily a part of the definition, which makes it difficult to place

[12] The stopwords were removed by the NLTK package (Bird, Loper & Klein, 2009)

the window at first. Another sub-optimal pre-processing step is that the word2vec model was built on data (context windows) not considering sentence boundaries. The context window in SemDaX and Korpus DK do not extend sentence boundaries. This situation slightly influences the word embeddings, but probably not greatly, if each part of the sentences not in interest, contains an equal amount of white noise, or is simply – and reasonably - considered as more context.

In summary, after pre-processing, each sentence or bag-of-words becomes a vector representing that instance in the word embedded space.

4.1 From Dictionary Label to Synset id

The evaluation data SemDaX is, as written above, annotated with dictionary senses. A key from these senses to the corresponding senses in DanNet is necessary to bootstrap the senses created from DanNet.
For each target noun, and for each sense of these nouns, a DanNet synset id is found. For 17 target nouns 159 links are found, with an average on 9.4 senses. The DanNet granularity is slightly coarser than that of the dictionary. For this reason, some dictionary labels are linked to the same synset. See discussion of this in chapter 6.
Table 2 provides an overview of the sense numbers before and after the linking, the number of idiomatic expressions and the number of the senses apparent in the evaluation data.

Noun	**Senses in dictionary**	**Senses encountered in data**	**Idiomatic expressions**	**Senses after linking to DanNet Synsets**
Ansigt	22	16	9	**6**
Blik	9	8	2	**6**
Hold	11	10	2	**8**
Hul	25	22	9	**13**
Kort	22	21	13	**10**
Lys	33	30	18	**16**
Model	9	9	1	**8**
Plade	20	13	3	**13**
Plads	25	21	9	**10**
Skade	16	12	7	**6**
Slag	32	28	13	**15**
Stand	17	11	7	**4**
Stykke	33	22	6	**16**
Top	14	12	6	**5**
Vold	16	10	2	**7**
Kontakt	10	9	0	**7**
Selskab	11	11	1	**9**

Table 2: Overview of sense numbers in the dictionary (annotation options), the senses used in data (chosen annotations), the idiomatic expressions in the chosen annotations, and the resulting number of senses when linked to DanNet.

Several decisions need to be made in the manual data linking. Firstly, numerous times the dictionary labels represent a figurative sense, especially when used in an idiomatic expression.

Most of the dictionary labels, that do not have a direct corresponding DanNet synset, are of this kind. As the evaluation data contains many idiomatic expressions, it would be a waste of data to leave these instances out. Therefore, the dictionary labels of the target noun in the figurative expressions are merged with the synset that corresponds to the literal sense of the noun. An example is '*kaste lys over*' ('*shed light on*'), where the noun '*lys*' ('*light*') is used figuratively for '*attention*', '*awareness*' or the like. In such cases the literal sense "within" the metaphor is chosen. This also follows the principle of annotation of idiomatic expressions or other figurative speech in the before mentioned work of Pedersen et al. (2018) where the annotation of the SemDaX data is described, and also used in a WSD task. As this thesis follows the same principle, a fair comparison of the two methods is more reachable in future development.

Secondly, if a dictionary label for a given target word does not have a direct corresponding DanNet synset (of the same word form), then a synonym or near-synonym of the target word is used. It is preferable to find a synonym or very-close synonym than pointing to the same synset more than necessary to get the variety of dictionary labels represented most accurately, and to save as much data as possible. An example is the noun '*slag*' (battle, stroke, cape, roll, beat), understood as in a roll of a dice. This word sense was not directly found in the entries of the possible DanNet synsets under '*slag*' – but the synonym '*terningekast*' (dice roll) was found, and is therefore chosen as corresponding synset. While linking from dictionary senses to DanNet synsets of these target word samples, it was never the case that DanNet was more fine-grained. DanNet has been semi-automatically compiled from the Danish dictionary, so the number of senses generally corresponds (not taking idiomatic expressions into account).

Thirdly, 12 synsets do not have a given sample sentence in DanNet. Instead, sample sentences are retrieved from the website of the Danish dictionary[13], *Den Danske Ordbog*, for each corresponding sense. In one case, the definition of the word sense is filled in, as no sample sentence is given.

The key is available at the GitHub page[14] of this thesis work, where more examples and comments on each link is given. The key from dictionary labels to DanNet synsets is a necessity for the approach of this thesis. The details of the approach and motivation for the four experiments of this work are given in the next sections.

4.2 From Word Embeddings to Sense Embeddings

Since the word embeddings carry information about similarity of words at word level, the word sense embeddings need to be built to yield similarity information at sense level. This can be done in various ways (see also section 2.1.5). The main idea in this thesis work is to find a sense representation of each DanNet synset of each of the target nouns, since the labels in the evaluation data can be linked to DanNet synset id's. The sense representation is made with

[13] https://ordnet.dk/ddo/ordbog?query=ansigt Retrieved 10.05.18
[14] https://github.com/idaroermann/danish_nouns

various information extracted from and with DanNet (experiment 1-3). As mentioned in the introduction, the quality of the sense representations is evaluated in a WSD task on the evaluation data SemDaX. Finally, the robustness of the representations found in experiment 3 are tested in a machine learning classifier.

The idea to extract knowledge-based information from DanNet to use in combination with the distributional word sense in the word embedded space is not only motivated by the availability of materials. As presented at the end of 2.1.2, wordnets contain hand-picked words, that play important semantic roles to every concept, to every synset. This also follows e.g. Pustejovsky's theory on Qualia Structure, that argues that the meaning of a word can be found by examining the kind of relation the relevant surrounding words have to the target word. If the synset members in DanNet are important to the concept's use in language, then the collections of those words in a distributional model as the word2vec model, is reasonable, since such a model models word similarity based on the distribution of words used in data. If there is a correspondence between how the high-quality knowledge-based information in DanNet is distributed and how the distributional information of the words looks in the vector space model, it could be useful for creating sense representations on the basis of the given lexical resource. When sense representations (for sense-tagging) are built independently of semantic annotated evaluation data, the scope of the approach reaches all the words contained in the input resource, and not only those words encountered in annotated data.

4.2.1 Experiment 1: Sense Embeddings from Synset Members

Input: DanNet synset information (as words) + word embedding model
Output: One sense embedding per word sense per target noun

The idea of the first experiment follows Bhingardive et al. 2015 (see 2.1.5) where the wordnet synset neighbours (members) are collected in a "sense-bag". The sense-bag of each synset consists of the nodes linking to the synset by different semantic relations (hypernyms, hyponyms, synonyms etc.), as well as the sample sentence (the word used in context). In this thesis work, the definition of the sense is added to the sense-bag as well. The mean of the word embedding vectors (from the word2vec model) of the words in the sense-bag are taken, and that new vector is the sense embedding of that given synset. In this way, all the words associated with the target word in the wordnet are put together, and the sense embedding for the given synset is the mean of the corresponding vectors in the word2vec model.

Though the task of Bhingardive et al. 2015 work (to find most frequent sense) is different than the task of this thesis work (WSD), the argument for representing senses this way is the same. The semantic relations linking to the synset members from the word in the middle are words describing main senses, examples of the senses, the domain, what they are used for, what they are made of etc. For example, consider the word '*model*'. '*Model*' can have something to do with a mathematical representation (as a vector space model or climate change prognosis), as well as with the employed person working in fashion at a photoshoot. The words associated with the former sense are more similar in meaning than to the words associated to the latter sense, which again has more in common in the contexts they are used in. If the associated words in the two groups mostly are used in two significantly different contexts, then their created sense embeddings would probably reflect that in the word2vec model. Experiment 1 is therefore a test of whether the selected knowledge-based information from DanNet in combination with the distributional representation of the words in the sense-bags can construct an appropriate sense representation.

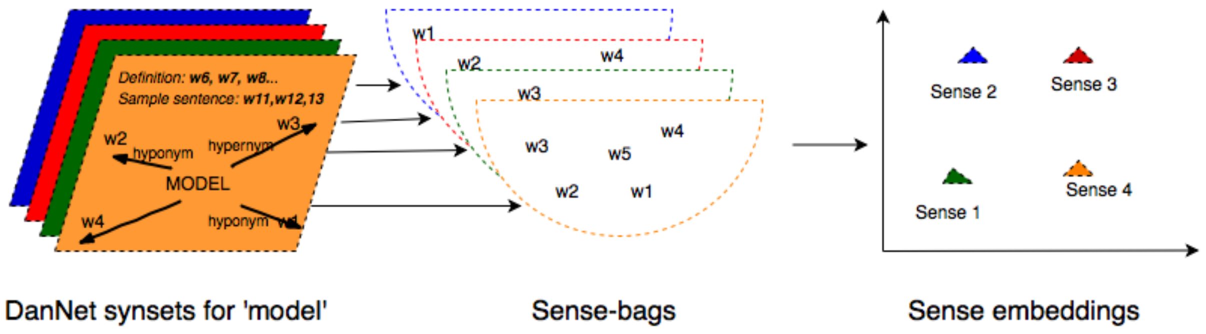


Figure 7: Visualization of experiment 1. The squares represents DanNet synsets, the half-circles represents the sense-bags where selected information from the synsets are stored, and finally a 2D example of how each sense-bag is represented in the word embedded space.

More formally, a sense-bag is a set, $B = \{w_1, .. w_n\}$, where n is the number of words in B, and the w's are the words selected to be considered. Each word, w_i, in B, can be represented by a vector $\vec{W}_i$, using the word2vec model. From these vector representations, a mean vector, $\vec{M}$, can be calculated by

$$\vec{M} = \frac{\sum \vec{W}_i}{n}$$

A reason to take the mean of the word vectors is to treat each word equally, and to keep the length of the vectors. See discussion on this matter in chapter 6.

The information extracted from the wordnet to put in the sense-bag to create the sense embeddings comes from different sources, as mentioned above. In experiment 1 several combinations of information are tested. The different components are shown in Table 3.

(1) Local synset members
(2) Hypo- and hypernym synset members
(3) Sample sentence
(4) Definition
(5) The definitions in synsets of hypernyms and hyponyms

Table 3: The five components that are extracted from DanNet synsets.

The results in the following experiments will be compared to the WSD performance of the best sense embeddings, namely those created with (1), (3), (4) and (5).

4.2.2 Experiment 2: Sense Embeddings from Synset Sample Sentence

Input: DanNet synset sample sentence (as words) + word embedding model
Output: One sense embedding per word sense per target noun

The nature of this second experiment is similar to that of the first experiment. The idea, though, is to exclusively use the sample sentence given in DanNet to create the sense embeddings. As before, the words in the sample sentence are collected in a sense-bag, and the mean of the word2vec vectors in the sense-bag is taken. The motivation is to see how well a sense representation made solely on the basis of the target word used in a context (as vectors) corresponds to the instances (as vectors) in the evaluation data. On the one hand this could work well, since the DanNet sample sentences are hand-picked by lexicographers who consider the sample sentence as a good example, and since the two instances that is compared (the sample sentence and the test instance) is both made of bag-of-words of a sentence where the given word is used. On the other hand, it could work less well than experiment 1, as more hand-picked associated words are gathered in the sense-bag, which could reinforce the right direction of the sense embedding in the word2vec model.

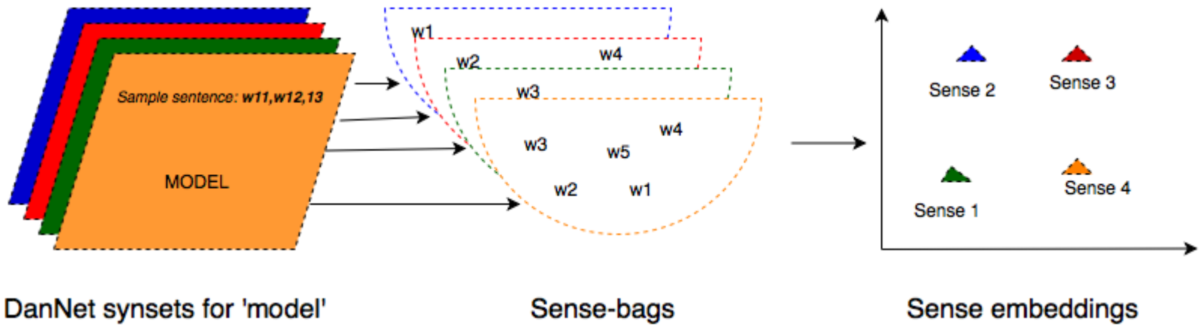


Figure 8: Visualization of experiment 2. The squares represents DanNet synsets, the half-circles represents the sense-bags where the sample sentence from the synsets are stored, and finally a 2D example of how each sense-bag is represented in the word embedded space.

Formally, experiment 2 is like experiment 1, where the sense-bag is a set, $B = \{w_1, .. w_n\}$, where n is the number of words in B, and the w's are here the words *exclusively* originating from the sample sentence. Each word, w_i, in B, can be represented by a vector $\vec{W}_i$, using the word2vec model. From these vector representations, a mean vector, $\vec{M}$, can be calculated by

$$\vec{M} = \frac{\sum \vec{W}_i}{n}$$

As in experiment 1, this procedure creates one single sense embedding in the vector space per word sense per target noun. These sense embeddings are later used to disambiguate the test sentences. The creation of the sense embeddings in experiment 3 is described first, before details on the WSD method is given.

4.2.3 Experiment 3: Sense embeddings by cluster centroids

Input: Sense embeddings from exp. 2 + unlabelled Korpus DK instances (as embeddings)
Output: One sense embedding per word sense per target noun,
Sense-labelled Korpus DK instances

The idea of experiment 3 is to re-use the sense embeddings found in experiment 2 (created from the wordnet sample sentence), and then tune the sense embeddings by adding more data than merely information from one sample sentence. The new data is unlabelled instances (containing the target words) which is clustered with the k-means algorithm (see section 2.2.1 for computational theory). Each cluster is evolved from a given seed, namely the (labelled) sense embeddings from experiment 2, which bootstrap the clusters to a category. In this way, the location of the centroids (initially the seeds) in the vector space is adjusted when given more data, and the un-labelled instances are hereby auto-tagged. The sense embeddings created in this experiment are the centroids of the clusters.

The motivation for adding more data is to avoid the sense embedding exclusively to be based on the single sample sentence. It is to test whether the fine-tuned sense embeddings represent the senses more similarly to the human annotated evaluation data, than just the sample sentences alone in experiment 2, or the synset members and neighbours in experiment 1. Yet a reason is to see if the raw data can be clustered on the basis of the seeds at all, in a way that can perform WSD. If this experiment works well, it can be used for bootstrapping more semantically tagged evaluation data.

The new data is from Korpus DK, described in section 3.5. Every sentence containing the target words is extracted, and a context window of 5 words to the left and right of the word is chosen. For each target word around 1000 instances is considered.

Each sense embedding from experiment 2 is used as a seed in the k-means algorithm applied with the software package SciKit-learn (Pedregosa & Varoquaux, 2011). Usually, the algorithm repeats the procedure of assigning data points to clusters with initial random centroids, until the same data points are assigned to the clusters. In this thesis work, the initial centroids are given, namely the seeds, so the procedure only runs once.

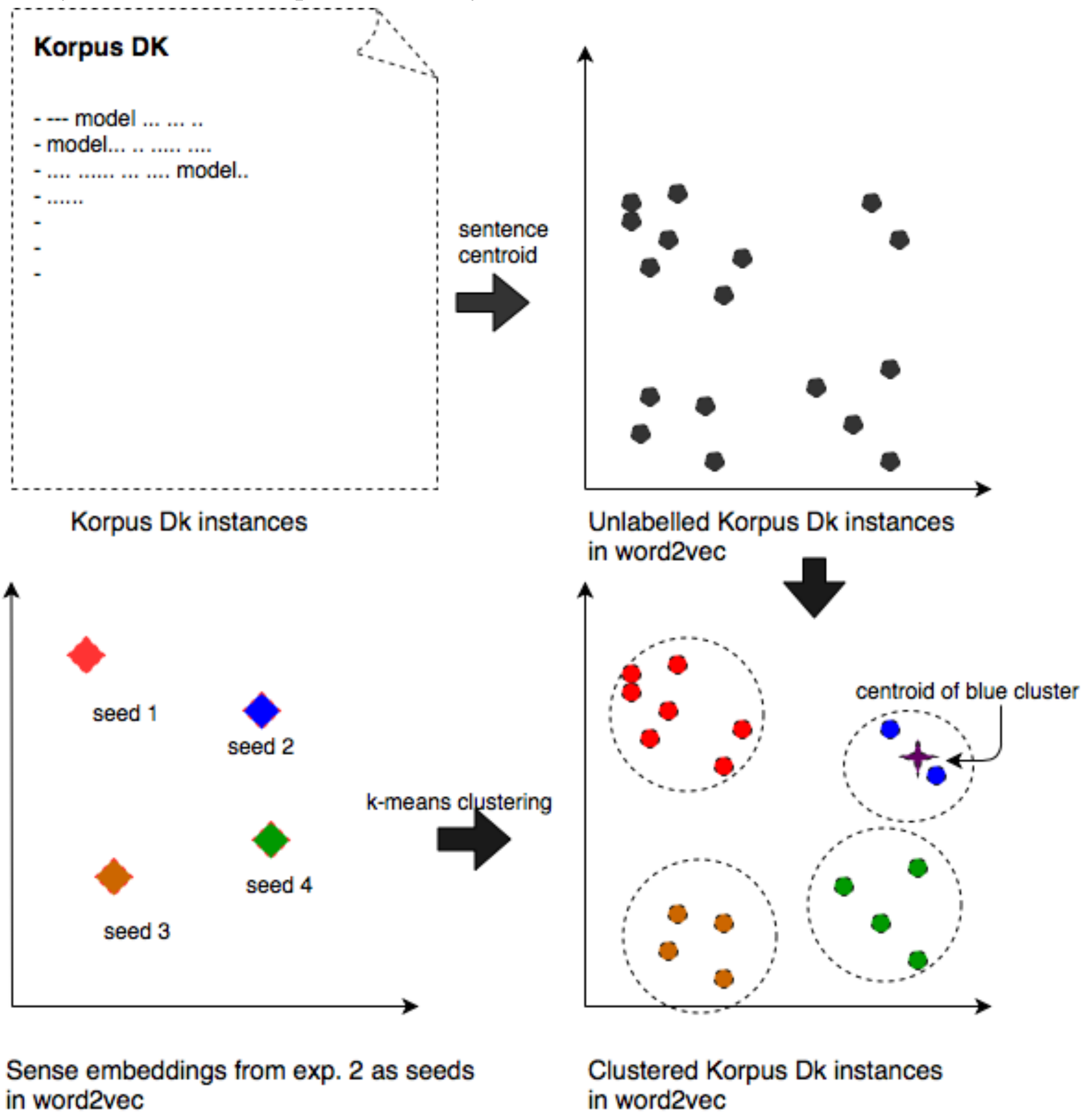


Figure 9: Overview of method of experiment 3. The unlabelled Korpus DK instances are represented in the word2vec space. These instances are clustered around the sense embeddings from experiment 2 with the k-means algorithm, resulting in clustered – labelled - Korpus DK instances. The centroids of these clusters are the sense embeddings created in this experiment.

The clustered Korpus DK instances for the target word *'top'* is shown in Figure 10 as a 2D representation. Each colour represents a word sense.

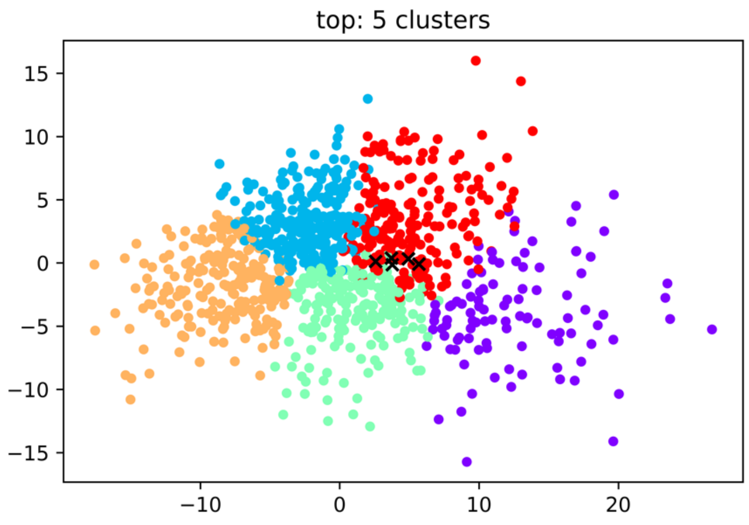


Figure 10: 2D representation of clustered Korpus DK instances for the target word 'top'. Dimensionality reduction with PCA. The black crosses in the central area are the original seeds. Plots for other target words shows a bigger distance between the seeds, though the clusters overlapped more. The plot for 'top' were chosen for explanatory reasons, not for analysis.

The centroids of the clusters are the sense embeddings created in this experiment. The new labelled instances from this experiment are further used and tested in a machine learning classification task in the final experiment. First, the evaluation method of experiment 1-3 is presented in the following section.

4.3 Evaluation: Word Sense Disambiguation task

Input: Sense embeddings from experiment 1-3
Output: One sense prediction per test instance

The method of evaluation in these experiments is WSD. The reason is that the data available for Danish semantic NLP at this stage is sense annotated sentences and super-sense annotated senses, but not ranked senses, as used in the before mentioned SemEval 13 task, or sets of relevant words, which is suitable for unsupervised WSI evaluation. The scope of this thesis of course also limits the possibility to create new evaluation data. See alternative evaluation methods in 2.1.6 and discussion on this choice in chapter 6.
The quality measure of the system made in this thesis, is how good the system is to word sense disambiguate the evaluation data SemDaX. As stated in the beginning of this chapter, all the sentence instances are represented as vectors. All the senses each instance can have, is also represented by a vector: One vector for each sense of each target noun. The disambiguation

procedure is simply to compare the sentence vector to each possible sense representation, and let the system choose the most similar sense representation to the sentence vector. A distance measure to compare similarity needs to be chosen. The similarity measure used, D, is the cosine value to the angle between the vectors:

$$D = \cos\theta = \frac{A \cdot B}{\parallel A \parallel \parallel B \parallel} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

where A and B are the vectors to compare, θ is the angle between the A and B, and n the dimensionality. The smaller the angle is, the higher similarity. The similarity (or distance) measure is usually the distance between the endpoints of the vectors, the dot product, the angle or the cosine between the angles. When choosing cosine to the angle, then the lengths of the vectors are outbalanced, and the value is kept between 0 and 1.

The results of the WSD are compared to the performance of experiment 1, the most frequent sense, as well as the random choice.

The system computes vectors representing senses and sentences. Now, an example is given in words, and not vectors, to simulate what the WSD task would look like if a human were to disambiguate with the same information as the system has in vectors.
The target noun '*model*' (roughly same word and senses as in English) has 9 senses encountered in the data and 1 idiomatic expression. See Table 4. The words in bold are not very common words and therefore those which probably influence the location of the sense embeddings the most. The idiomatic expression '*stå model til*'[15] has been collapsed with sense 5, as the closest in meaning, since idiomatic expression generally do not appear in wordnets.

Three test sentences from SemDaX for the word '*model*' are shown below.

- ***"Og så havde vi kursister den luksus også at have fire fantastiske modeller at arbejde med"***
  (*"and we as participants then had the luxury of having four fantastic models to work with"*)

- ***"Men sådan er prisklassen konkret, og de fleste modeller bliver ofte kun produceret i et meget lille antal"***
  (*"But that is how the price level actually is, and most models are produced in a very limited amount"*)

- ***"Jeg bryder mig ikke om ordet model"***
  (*"I do not like the word model"*)

Which senses would you choose from Table 4, if you only consider the last column, and mainly the bold words? Note, the words should be considered out of order as in a bag-of-words.

[15] Direct translation: "*stand model to*" . Means to put up with/stand for/be the one to accept something. 'model' refers here to the hypothetic person who accepts something.

Sense of ‘model’	DanNet sample sentence	Synset members
1. Representation of something (sometimes on a smaller scale)	***Færgen** er en **model** i 1:4* The **ferry** is a **model** 1:4	*Effekt, videnskab, fremstille, figur, afprøve, gengive, pynte, arbejdsmodel, gine, globus, globus, mockup, modelbygning, modelfly, skalamodel, skibsmodel, modeljernbane, modelbil, modelskib, modeltog, kirkeskib* Effect, science, produce, figure, test, represent, decorate, working model, gine, globus, mock-up, model building, airplane model, scale model, ship model, traintrack model, car model, ship model, train model, church ship
2. Form of procedure, method or norm	***Social-** og **sundhedsforvaltningen** har **udarbejdet** en **model** for **løsningen** af de **administrative opgaver*** The **administration** of social and **health affairs** has **prepared** a **model** to **solve** the **administrative tasks**	*Fremgangsmåde, psykologi, automat, løsningsmodel, opskrift, skabelon, analysemodel, beregningsmodel, budgetmodel, finansieringsmodel, fortolkningsmodel, identifikationsmodel, samfundsmodel, standardmodel, styringsmodel, udviklingsmodel* Procedure, psychology, automat, model to find solutions, recipe, model for calculation, model for making budgets, model for financing, model to interpret, model for identification, solution model, calculi, budget model, financing model, interpretation model, identification model, standard model, model for leading, development model
3. Type or edition of a certain product	***fabrikkens** største **model** D 9000 Jumbo .. er **Europas største konventionelle mejetærsker*** The biggest **model** of the **factory**, D9000 **Jumbo**, is **Europe’s biggest conventional combine harvester**	*Handelsvare, industri, handlende, køber, sælge, købe, bordmodel, demonstrationsmodel, produktionsmodel, slimlinemodel, unisexmodel* Goods, industry, dealers, buyers, sellers, buy, table model, demonstration model, production model, slimline model, unisex model
4. Schematic description or illustration of an abstract, complicated thing or relation	***Watson** og **Crick fremsatte** deres **model** af **DNA-molekylet** som en **dobbeltspiral**, der kan **visualiseres** som en **vredet stige*** **Watson** and **Crick presented** their **model** of a **DNA molecule** like a **double-spiral**, that can be **visualized** like a **twisted latter**	*Anskueliggørelse, videnskab, atommodel, forklaringsmodel* Visualization, science, atom model, explanation model
5. A person who poses for a photographer or painter or sculptor + idiomatic expression ”stå model til”	***Edwards flyttede** ind hos **Bacon**, der **benyttede** ham som **model** i mange af sine **højt betalte malerier*** **Edwards moved** in at **Bacon’s place**, who used him as a **model** in many of his **highly-valued paintings**	*Person, kunst, tale, tænke, leve* Person, art, speak, think, live
6. A person who is employed by a company to wear its products	***Erik Mortensen** førte selv **bruden** frem på **podiet** til **klapsalver** fra både de øvrige **modeller** og **publikum*** **Erik Mortensen led** the **bride** to the **stage** under **applauses** from the other **models** and **audience**	*Person, reklame, job, tale, tænke, leve, mannequin* Person, advertisement, job, speak, think, live, mannequin
7. Person, object or phenomenon that functions as a role model	*Den **barmhjertige samaritan**, der tager sig af de **nødstedte**, er blevet en **model** for den gode **handling*** The good **Samaritan**, who takes **care** of the **people** in **need**, has become a **model** for the good **choice**	*Person, psykologi, tale, tænke, leve, rolle, aktantmodel* Person, psychology, speak, think, live, role, actantial model
8. Person who (professionally) let themselves be photographed	*Hun har været [..] **topløs model** på **britiske vodkaannoncer*** She has been [a] **top-less model** on **British vodka-adds**	*Person, foto, job, tale, tænke, leve, topmodel, nøgenmodel* Person, photo, job, speak, think, live, top-model, nude model

Table 4: Information from DanNet for the noun ’model’. The information corresponds to what the system gets as input to solve the WSD. The table shows the possible senses for ’model’, the DaNNet sample sentence, and the English translation. Bold words are considered most influential (as being neither function words nor very common words). The system is only presented with the last column in bag-of-words and builts a sense embedding for each possible sense.

4.3.1 Evaluation statistics

Two evaluation measures are used to estimate the performance of experiment 1-3, namely the accuracy score and the Kullback-Leibler divergence (KL divergence) score.

Accuracy

The accuracy score is a simple measure to show how correct the system guesses on average. The accuracy is the count of correct disambiguations, T, out of the total count of instances, *N*:

$$\text{Accuracy} = \frac{T}{N}$$

As the annotators of the evaluation data, SemDaX, sometimes disagree, there can be several annotated sense tags in the data. This means that there possibly are multiple (correct) classes per instance, as every instance is considered correct in this setting. The annotators are not ranked, nor are e.g. the most frequent annotation picked per instance. A reason for low inter-annotator agreement is many of the sentences lack context to fully tell which sense is used[16]. Therefore, all the annotations are saved, also to reflect the disagreement among annotators. This means that there might be more than one correct answer (annotation) per instance, which gives the system an advantage. The accuracy score results therefore need to be analysed with the inter-annotator agreement, which will be presented alongside the results.

An F-score is not directly accessible. There are several classes per instance, and the combination of classes might change at every instance. The accuracy score can usually be calculated from finding the true positives (TP), true negatives (TN), false negatives (FN) and false positives (FP). To find these sets, one class needs to be fixed. For every instance, that fixed class can be truly guessed, truly not guessed etc. But as each possible class is considered correct in the evaluation setting here, the size of TP, TN and FP can be misleading, which affects the calculation of precision and recall. A FP might actually, in this setting, belong to a "TP", if that guess is among the correct guesses, though not being the fixed class. Similarly, a TN might be a TP. For example: we can fix the class A in the set of classes *A, B,* and *C.* The correct guess is A and *C,* and the system guesses *C.* This guess will usually be a FN, if A is encountered as the only correct class. But with several correct classes, then C should be encountered in a set of True's. A possible work-around is to represent the annotations of each instance in binary form: a matrix of all the instances and their annotations per sense as one-hot vectors can be compared to a similar matrix representing the guesses by the system. This would again lead to a biased score as the system only guesses one class, but there are several correct answers in the annotation matrix, which would lower the similarity between the two matrices. Nevertheless, it is possible to calculate whenever the system guesses incorrectly or capture one of the possible classes, which is why the accuracy score is considered in this work.

[16] Another reason is the rather fine-grained sense inventory

Kullback-Leibler divergence

The KL-divergence is also used as an evaluation measure. This is to incorporate the fact that the guesses by the system can be more or less precise: the predictions can be weighted for all classes per instance, as some senses are more similar to the guess than other. This measure will also assist the abovementioned difficulty where there are several correct classes, but only one prediction.

As described in 2.1.7, KL-divergence is usually used to compare probability distributions. In this work, the KL-divergence is used to compare the distribution of the distances between the built sense representations and the test instances ('distance distribution'), to the distribution of annotations of the test instance ('annotation distribution'). That is to take the KL-divergence of the normalized score from the normalized distribution represented by the ground truth (the annotations).

The distance distribution is the cosine similarities, from each sense representation (built by the word sense representation system) to the test instance vector, converted to a probability distribution. The smaller distance (high similarity), the higher a probability value.

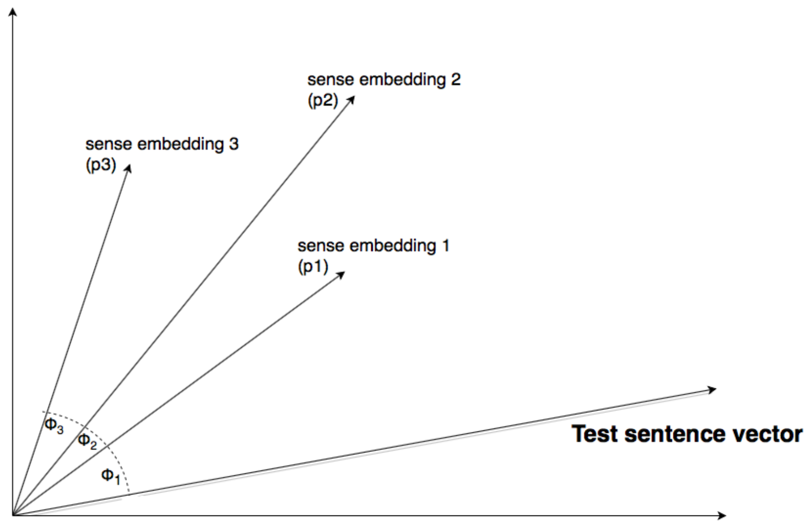


Figure 11: The distribution of sense embeddings along with the test instance vector. The smaller angle between the senses and the test instance, the higher a probability score, P.

The procedure to determine the distance distribution starts by calculating the cosine similarities, D_i. These measures are then transformed to all lie between 0 and 1 using

$$D_i' = \frac{1}{2}D_i + \frac{1}{2},$$

where, D_i', is the transformed measure.

The sum of all D_i' are then calculated, $S = \sum D_i'$, and the frequency, p_i , is calculated:

$$p_i = \frac{D_i'}{S}$$

The set of p_i is used as an estimate of the probability distribution, where the most similar sense representations (to the test instances) are assigned the highest probability score.
As the probability values are the normalised distances from the test instance to each of the sense representations, the probability values can never be 0: there will always be a distance in the geometric vector space (unless if, hypothetically, the test instance representation and the sense representation is identical).

The annotation distribution is the annotations converted to a probability distribution. The frequencies, q_i , of annotation of each sense, a_i are calculated by

$$q_i = \frac{a_i}{\sum a} + \varepsilon$$

where a small value, ε, is added to avoid division by zero when calculating the KL-divergence below.

The KL-divergence, D_{KL}, is given by

$$D_{KL}(q||p) = \sum_{i=1}^{N} q(x_i) \cdot log\frac{q(x_i)}{p(x_i)},$$

where *p* is the probability distribution ('distance distribution'), *q* is the approximating distribution ('annotation distribution'), and *N* is the size of the distributions (number of senses). In other words, it calculates how far the true outcome (annotations) are from the output of the system (built word senses). If $D_{KL} = 0$, then *p*=*q* , which means that the two distributions are identical. For this reason, the lower a KL-divergence, the better a fit for the system to human annotations.
The situation where each built sense (p_i*'s*) has equal probability is used as a baseline.

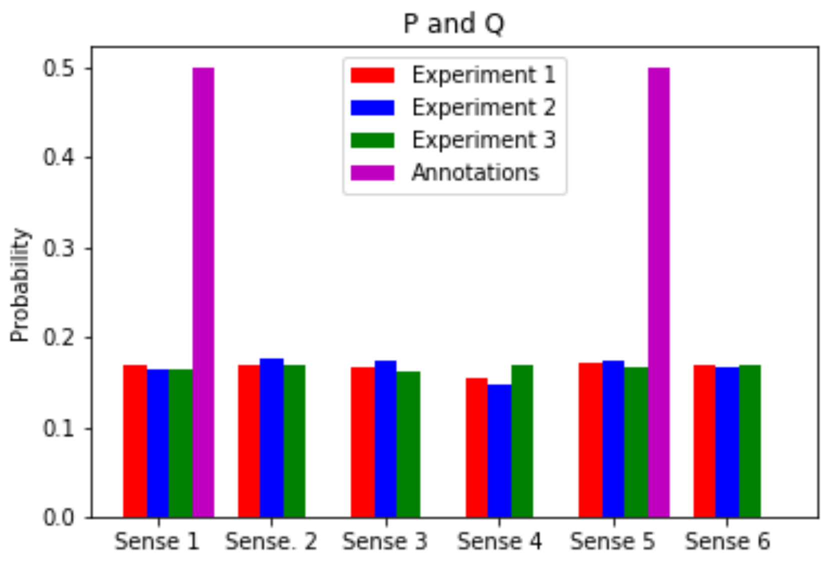


Figure 12: Probability distributions P ("distance" distribution) and Q ("annotation distribution") for the sentence "bliv oplyst om Facebooks sande ansigt her" ("be enlightened about the true face of facebook". The values in P are normalized distances from the induced word senses to the centroid of the test instance sentence. Both sense 1 and 5 is annotated, resulting in equal 0.5 probability per sense in the annotations (Q) in purple.

4.4 Classification Task for Machine Learning Algorithms

The last experiment is different than the previous ones as it does not seek to find a sense representation of the synsets, but rather a function from features to a sense label. The task is still WSD of the evaluation data, but this time with supervised machine learning classifiers.

4.4.1 Experiment 4: Classifier – SVM and FFNN

Input: sense-labelled data: (to train four individual classifiers)

(1) SemDaX
(2) Labelled Korpus DK
(3) SemDaX + labelled Korpus DK
(4) labelled Korpus DK (train) → SemDaX (test)

Output: One sense prediction per test instance (for all four models)

The idea of the final experiment is to train and test two machine learning classifiers (the traditional SVM and a simple feed-forward neural network) on the new-labelled auto-tagged Korpus DK instances from experiment 3, on the SemDaX evaluation data, on both data sets in a mix, and on labelled Korpus DK (for training) and SemDaX (for testing).

The motivation is to see whether it is possible for a machine to find and distinguish features of the classes in and across both data sets. If it is difficult to find a function from the data to predict a class, the data within each class is hard to distinguish, and therefore not distinct. If the clustering of the unlabelled instances from Korpus DK in experiment 3 does not separate the sense-classes similarly as the annotators did in the evaluation data, the function the classifier seeks to find is messier, and the results will be correspondingly less attractive. If the experiment works well, it would seem as if the locations of the seeds (sense embeddings) from experiment 2 match the centroids of the clusters from experiment 3, and that there is a correspondence to the sense annotations in the evaluation data.

Another motivation for this experiment is to compare feature selections with the previous WSD of the exact same evaluation data (Pedersen et al. 2018). Here, the features are the context lemmas of the sentence encoded as one-hot vectors, but not represented in the word2vec model. They also used the SVM algorithm, which makes room for reasonable comparison. The FFNN is tested as well to simply try how well deep learning as the state-of-the-art method solves the same task on SemDaX.

The SemDaX data is slightly modified for this final experiment. As annotators can disagree on which sense to tag with, and no ranking of the annotations are made, there can be multiple classes per instance. Following the same principle as the approach of Pedersen et al. (2018), the instance is repeated as many times as there is different annotations, and each repetition goes with a unique label from the set of possible annotation labels. This will undoubtedly be a function harder to find for the algorithm between features and classes. Nevertheless, if the machine is in doubt due to this fact, it corresponds well to the human disagreement on the same matter. Table 5 shows the size of the data before and after this procedure.

Noun	**Number of annotated instances**	**Number of repeated instances**
Ansigt	430	841
Blik	261	507
Hold	258	500
Hul	406	786
Kort	452	902
Lys	543	1055
Model	239	472
Plade	380	752
Plads	422	838
Skade	498	965
Slag	494	969
Stand	317	620
Stykke	549	1602
Top	263	523
Vold	325	637
Kontakt	175	345
Selskab	216	419

Table 5: SemDaX data size before and after repetition per class

Besides training and testing on the SemDaX (1), a classifier is also trained and tested on the cluster-labelled Korpus DK in itself (2). The motivation is to test whether there within each cluster are distinguishable features for the algorithms to identify and classify on the basis of.

Yet a classifier is trained from a mix of the two data sets (3). The motivation is to examine whether the class features are distinguishable for the algorithm, which would suggest a feature consistency – also across data sets.

Finally, a classifier is trained on the clustered Korpus DK data (4) , and tested on SemDax. The motivation is, as above, to examine whether the class features are distinguishable for the algorithm, which would suggest a feature consistency – also across data sets. The classifier might have a harder task here, compared to (3), as it has not seen data from SemDax at all under training. If the previous classifier performs well, this will supposedly work well too. If this is not so, this experiment will underline the differences of features within each sense class between the data sets.

The SVM

In this experiment, a Linear Support Vector Machine is built and trained using the module Scikit-Learn (Pedregosa & Varoquaux, 2011) from Python. The Scikit-Learn Dummy Classifier is tested as well, to access the performance by chance.

A k-fold cross validation is used to test if the classifiers work sufficiently, and is usually used to check whether features in the train/test set is representative for the whole data set. When data is split in train/test, and a machine learns the features in the training set, then predict classes in the test set, it is assumed that the training set is representative for the whole data set. If the features are significantly different in the test set from the training, it can explain a low performance.

A 5-folded cross validation is chosen, as the case is in the before mentioned work on SemDaX by Pedersen et al. (2018), that would be preferable to compare to. The reason to choose k=5, and not the usual k=10, is due to the size of the corpus: There is around 400 sentences for each target noun, and the classes to predict can be many. Hence, 5 folds gives more sentences in each fold. A low K decreases computational time, but a downside is that the classifier gets less training data.

The FFNN

Keras (Chollet, 2015), a neural network API with Tensorflow (Abadi et al., 2015) backend, is used to build the network.
The network is a simple feed-forward network visualized in Figure 13.

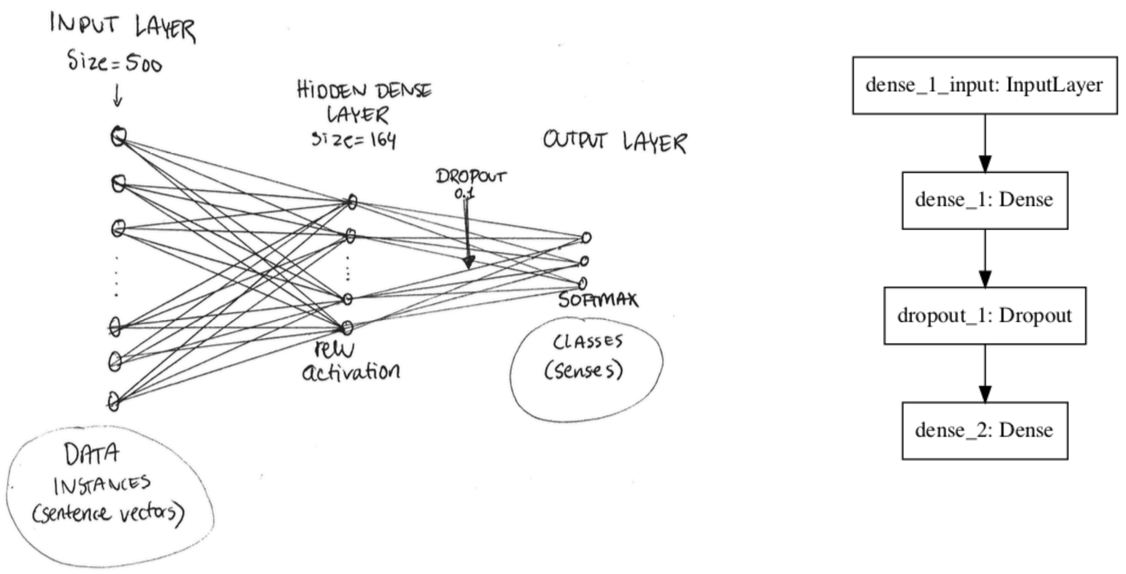


Figure 13: FFNN visualization. Input layer is ready for all the sentence vectors with 500 dimensions, a hidden dense layer of 164 nodes with rectifier activation. Then a dropout and finally the output layer with the size of possible word sense classes.

The vectors representing each training instance is given as input. Output is the class to predict for each instance. One hidden dense layer of size 164 with rectifier activation. A dropout of 0.2 to avoid overfitting, and finally the last softmax layer with the size of possible classes. Again, a 5-fold cross validation is applied.

In summary, there are two classifiers, trained and tested on SemDaX, the clusters from Korpus DK, on a combination of the two, and finally and most interesting one trained on the clusters and tested on SemDax.

5 Results

In this chapter we present the results. As mentioned, the sense embeddings created in the experiments are tested in a WSD task on annotated data.

In case of annotator disagreement, it is decided to treat each annotation equally, so that the WSD is counted as a correct instance if the guess of the system is among the annotations. The results presented in this chapter should therefore be compared to the table below showing the inter-annotator agreement across all words. Krippendorff's alpha, α, is applied. α = 0 means the agreement is not better than chance agreement, α = 1 means total agreement. Note, the nouns are highly ambiguous, so the usual requirement in annotation tasks of an α ≥ 0.80 is hard to reach here. The related work of Pedersen et al. (2018) finds an α ≥ 0.67 useful, which is mostly met in the agreement statistics:

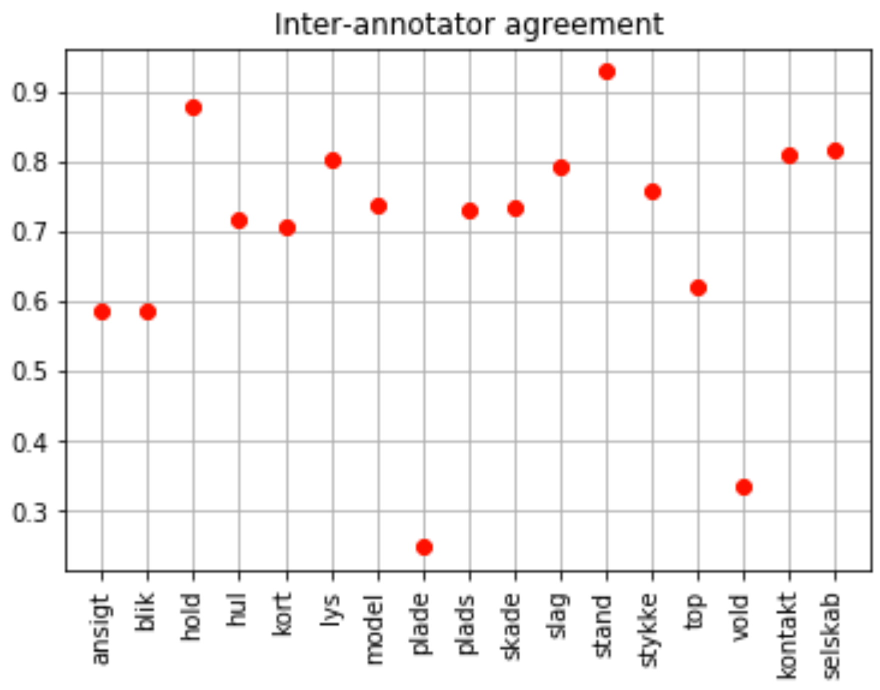


Chart 1: Inter-annotator agreement across all nouns of interest. Krippendorffs alpha, α, is applied. α = 0 means total disagreement, α = 1 means total agreement.

Experiment 1 to 3

The word sense representation method in experiment 1 extracts various wordnet information per word sense. The WSD performance of a selection of the various extractions (local synset members (local), hypo- and hypernym synset members (HH members), definition and sample sentence) are shown in Table 6. The best selection is kept as the main selection of experiment 1.

DanNet features	**Accuracy**
Definition	0.16
Local	0.22
HH definition	0.20
HH members	0.28
Local + HH definition + sample sentence	0.29
Local + HH definition + sample sentence + HH members + definition	0.29
Local + HH definition + sample sentence + definition	0.30

Table 6: WSD performance using various DanNet features. Note, only seven feature selections are shown, out of all the possible combinations.. The features with lowest performance and highest are shown,

Table 7 shows WSD results for the sense embeddings created with the word sense representation systems in experiment 1 to 3. The average random (by chance) performance is provided as well as the average frequency of the most common annotation.

WSD System	**Accuracy**
Experiment 1	**0.30**
Experiment 2	**0.27**
Experiment 3	**0.17**
Random	0.13
Most frequent	0.56

Table 7: WSD results. Accuracy score, random score, score of the most frequent annotation, and finally the results from the experiments compared to the other scores

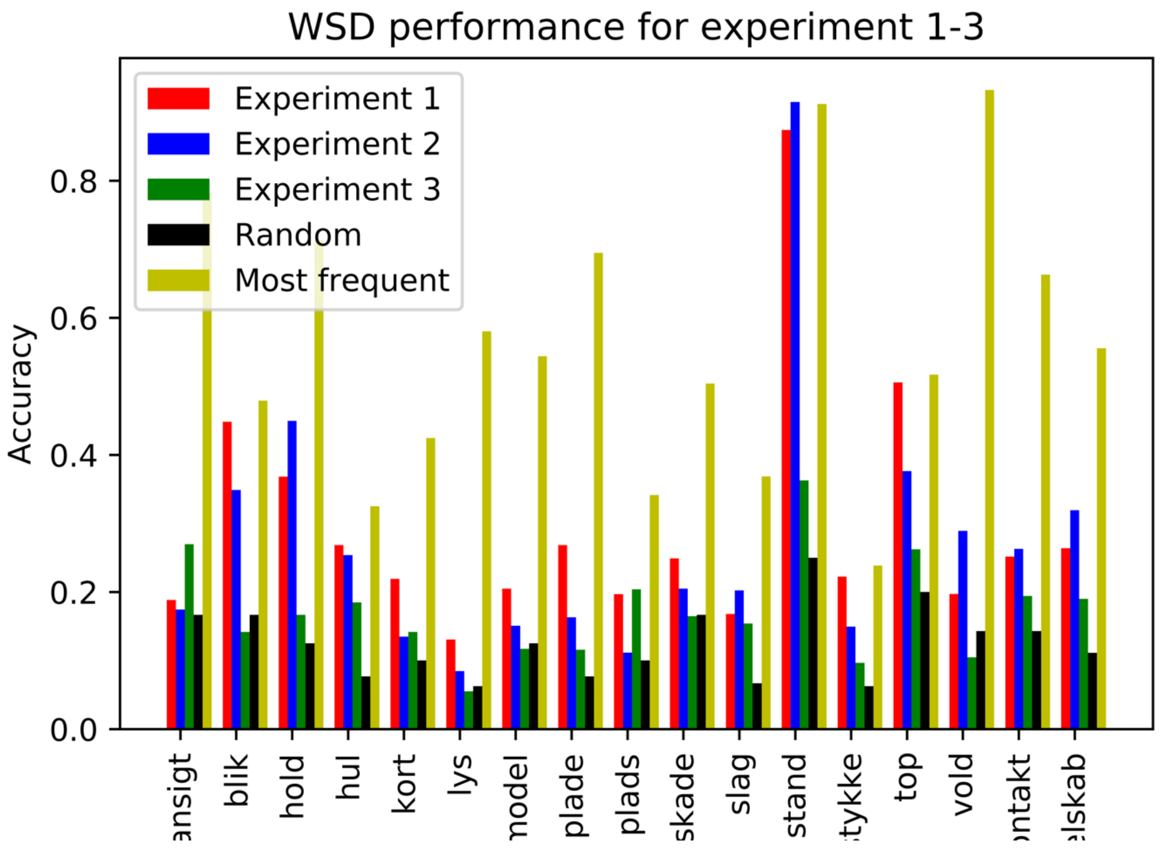


Figure 14: WSD performance for all experiments for all target nouns, as well as the random guess and most frequent sense frequency.

Figure 14 shows the WSD performance from experiment 1-3 for all words together with the random guess and most frequent. Remember, the sense embeddings in experiment 1 are made based on local wordnet synset members, neighbours, definition sentence and a sample sentence. The sense embeddings in experiment 2 is based solely on the sample sentence. The experiment 3 sense embeddings are the centroids of the k-means clustered Korpus DK sentences.

Figure 15 shows the average Kullback-Leibler divergence between the *distance distribution* (distribution of word sense representation system output) and *annotation distribution* (distribution in the evaluation data) for all test sentences for all words.

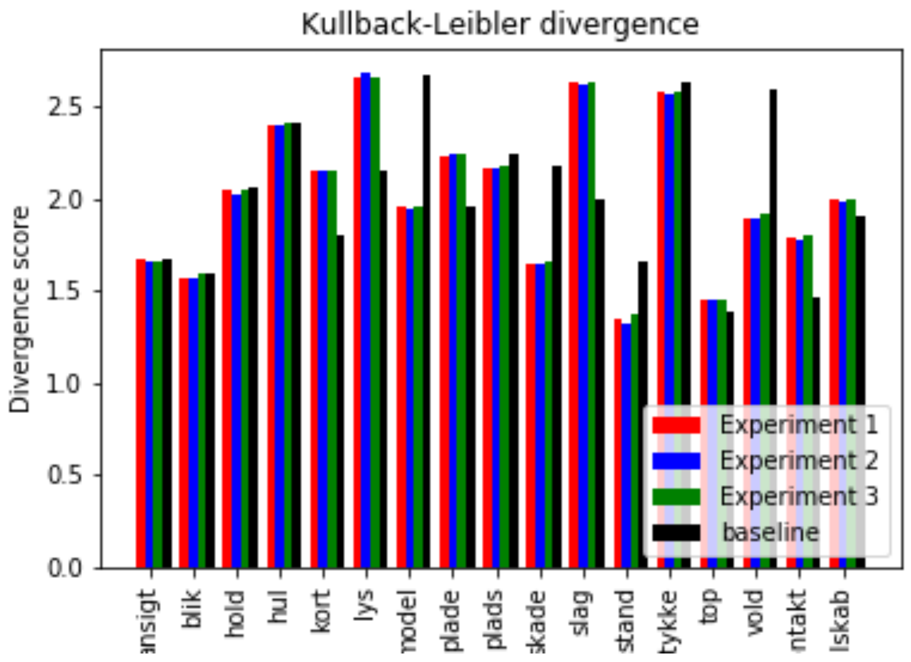


Figure 15: Kullback-Leibler divergence across all nouns and experiments. A divergence of 0 means perfect match between the compared distributions (not the case that a divergence of 1 is absolute dissimilar distributions).

Table 8 shows the average Kullback-Leibler divergence across all words for experiment 1 to 3. An epsilon of 1e^{-11} is applied.

Representation system	**Kullback-Leibler divergence**
Experiment 1	2.0100
Experiment 2	2.0054
Experiment 3	2.0198
Baseline	2.0220

Table 8: Average Kullback-Leibler divergence across all words. Baseline is the comparison with the annotations distribution from an artificial distribution where each sense is equally probable.

Experiment 4

Figure 16 shows the WSD performance for the machine learning classifiers trained and tested on the SemDaX data.

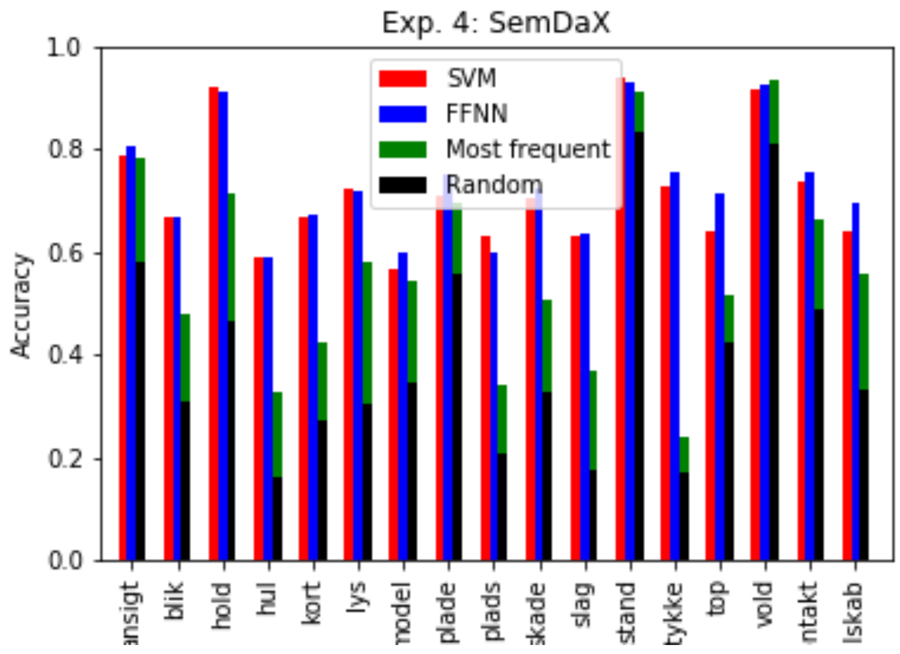


Figure 16: Machine learning classifier performance across all words. Trained and tested on the evaluation data SemDax. A Linear Support Vector Machine, a simple Feed-Forward Neural Network, and the Sci-Kit Learn DummyClassifier is applied for the random classification. The frequency of the most common sense annotation is given in green.

Figure 17 shows the WSD performance for the machine learning classifiers trained and tested on the k-means clustered Korpus DK data selection.

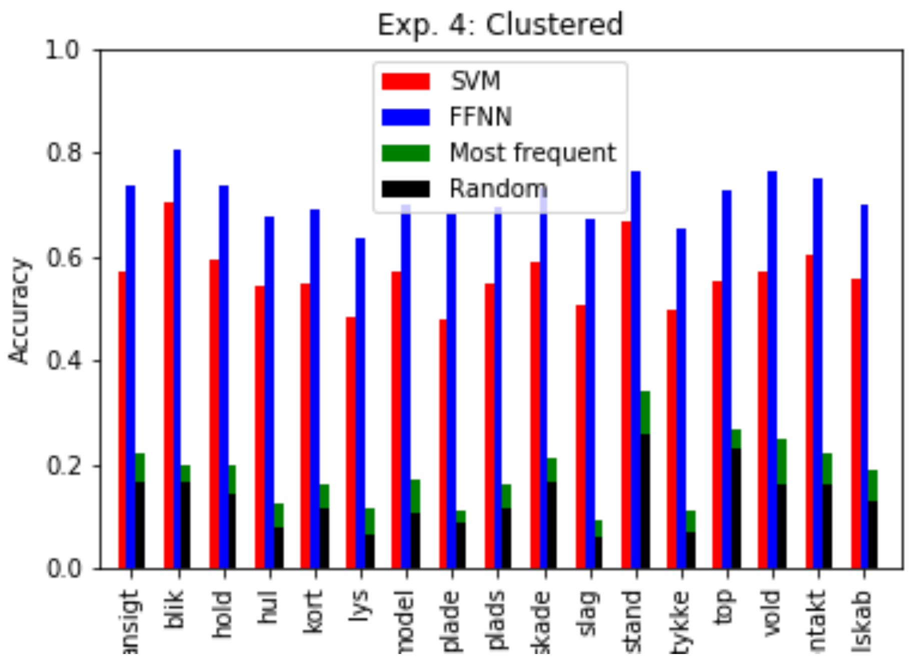


Figure 17: Machine learning classifier performance across all words. Trained and tested on the clustered Korpus DK data.. A Linear Support Vector Machine, a simple Feed-Forward Neural Network, and the Sci-Kit Learn DummyClassifier is applied for the random classification. The frequency of the most common sense (biggest cluster) is given in green.

Figure 18 shows the WSD performance for the machine learning classifiers trained and tested on the SemDax data combined with the k-means clustered Korpus DK data selection.

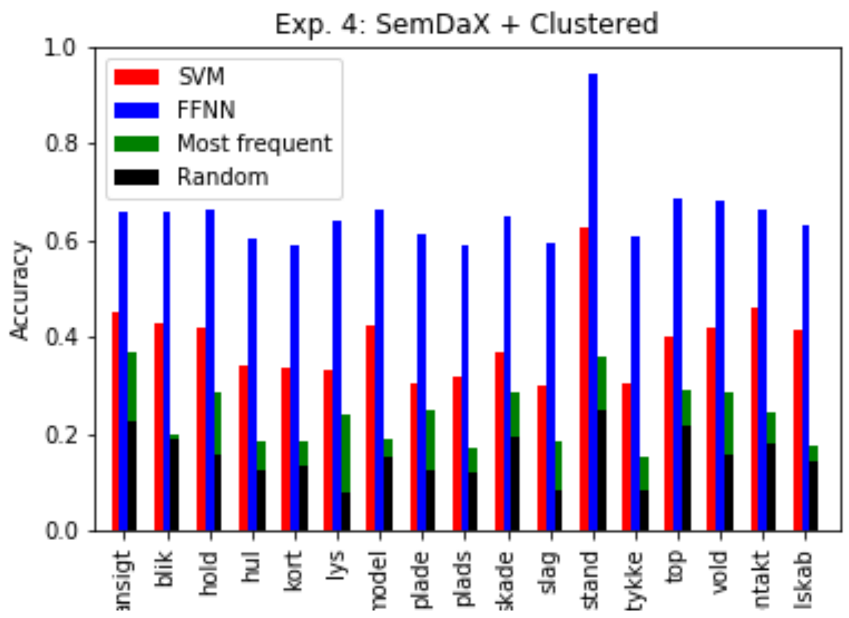


Figure 18: Machine learning classifier performance across all words. Trained and tested on the clustered Korpus DK data combined with the SemDaX data. A Linear Support Vector Machine, a simple Feed-Forward Neural Network, and the Sci-Kit Learn DummyClassifier is applied for the random classification. The frequency of the most common sense (annotation and biggest cluster) is given in green.

Figure 19 shows the WSD performance for the machine learning classifiers trained on the k-means clustered Korpus DK data selection and tested on the SemDax data.

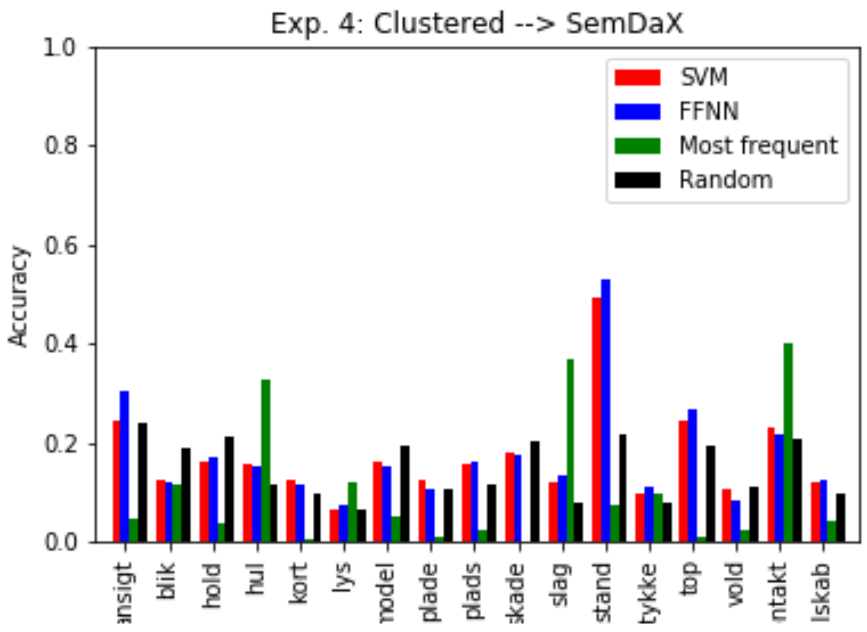


Figure 19: Machine learning classifier performance across all words. Trained on the clustered Korpus DK data and tested on the SemDaX data. A Linear Support Vector Machine, a simple Feed-Forward Neural Network, and the Sci-Kit Learn DummyClassifier is applied for the random classification. The frequency of the most common sense (frequency of class of biggest cluster in SemDaX) is given in green.

Table 9 shows the average performance of the machine learning classifiers in experiment 4.[17] There are no KL-divergence scores for this experiment: the score was computed from cosine similarities between sense embeddings to test instances in experiment 1-3. In experiment 4, machine learning classifiers are applied for WSD, not sense embeddings.[18]

Data \ Classifier	**Baseline (random)**	**SVM**	**FFNN**	**Most Frequent**
SemDax 5-fold	0.39	0.72	0.78	0.56
Clustered 5-fold	0.13	0.56	0.71	0.18
SemDax + Clustered 5-fold	0.153	0.39	0.65	0.24
Trained on clustered Tested on SemDax	0.147	0.17	0.18	0.10

Table 9: Average machine learning classifier performance for different data selections and algorithms.

[17] The evaluation method gives an advantage to the words with higher disagreement, as each annotation is correct. A classification of either of the possible annotation is a counted correct. The FFNN classifier *without this advantage* outperforms the SVM and almost reaches the performance of the FFNN with evaluation advantage.

[18] It would be possible to compute KL-divergence scores from the final softmax layer to the test instances considered as a probability distribution. This is indeed a relevant future project. (The test data has been slightly changed for this final experiment (repeated instances as many times there were multiple classes). The original test data had to be applied to compute the KL-divergence score for this experiment.)

6 Analysis and Discussion

In the first half of this chapter, an analysis of the results from each experiment is given. Both one-by-one, and across experiments. Afterwards, a discussion on the possible reasons for the system performance is given, especially with weight on the linking of the datasets, the evaluation method, the data applied, and the methodological assumptions taken in the approach.

6.1 Experiment 1-3

The experiments generally outperform the baseline of 0.13, but do not, as expected, beat the most frequent sense at 0.56. Experiment 1 outperforms the baseline for all target words with an accuracy score on 0.30. The DanNet information extracted for creating the sense embeddings that works best is when including the sample sentence, the definition, the local synset members, and the definitions from the hyper- and hyponym synsets. When extracting only one kind of information, e.g. the local synset members, it works less well. The general picture is that the single features performs worst, and the more features used, the better a performance (except with the HH members). Experiment 2 outperforms the baseline as well for all target words with 0.27. This is a bit lower than experiment 1. Experiment 3 also beats the baseline with an accuracy score on 0.17.

When comparing the KL-divergence scores across these experiments, the built senses are considered all together and compared to the objective, the annotations. All the experiments beat the baseline at 2.022, as these have lower KL-divergence scores. The results indicate that the built senses in experiment 2, with a KL-divergence of 2.0054 is a better match to the annotations, than experiment 1 and 3 with KL-divergences on 2.01 and 2.0198, respectively. This suggest that even though experiment 1 performs better at the WSD task, the sense representations in experiment 2 is a better fit to the true outcome of the system when looking at all the built senses together. Although, the significance of these differences can be questioned, and must be answered in future work.

For the word '*top'* and especially '*stand'*, the performance of experiment 1 is higher than the other words. This might be due to the low number of senses of these words: '*stand'* has 4 senses, and '*top'* has 5, where the average sense number is 9.4. Also, *stand* is often annotated with the same sense (also high inter-coder agreement) which suggests that there is a highly dominant sense both created and annotated: if a sense is very commonly annotated, and generally has highest cosine similarity with the test sentence, the performance will undoubtedly increase. The same situation might be the case in experiment 2 where '*stand'* also has a high score, and actually on the same level as the most frequent sense. This situation is also reflected in the KL-divergence score for '*stand'*, which is lower than the other words, suggesting that the created sense representation per sense for this word match the annotators' choice.

In experiment 1, the WSD of *'blik'* also works quite well compared to the other words considering the performance of the most frequent sense. This word has a relatively low inter-annotator agreement and “only” 6 associated senses, which could explain this. This word is also a homonym, which could increase the distance between the sense embeddings in the word embedding model.

Experiment 1 generally outperforms the other experiments, suggesting the highest correlation of the word uses per sense in the word embedding model and in DanNet (and how the test instances appear of course). Where experiment 1 works less well than experiment 2 is especially for ‘*hold'*, and ‘*vold'*, but also for ‘*slag'*, ‘*stand'*, ‘*kontakt'*, and ‘*selskab'*. Investigation of the synset member size for ‘*hold'* shows that almost half of the synsets only has one concept associated with it in DanNet, namely one hypernym. This is rather little information to base a sense embedding on, and, hypernyms tend to be more general, which might not be very informative being the only information.

‘vold' has a very low inter-annotator agreement and high most frequent sense frequency, which could give the system an advantage. Again, for this word, many of the synset sizes are rather small, which could explain the low performance compared to experiment 2.

‘selskab' is a tricky word as the differences in senses are often very subtle. Though the inter-annotator agreement is very acceptable, and therefore not giving the system a big advantage, many of the synsets in DanNet share quite some associated concepts. This could suggest that the positions of the sense embeddings for ‘*selskab'* in experiment 1 are close.

The experiment 2 sense embeddings perform slightly worse than experiment 1 in terms of accuracy score. This could be due to more noise in the associated words, as these are the bag-of-words of the whole sample sentence. However, the sense embeddings from experiment 2 works relatively good for the homonyms ‘*hold'* and ‘*vold'*. Their contexts are relatively different when looking at the sample sentences giving a hint on which group of senses of the homonyms that are met. The sense embeddings for ‘*selskab'* also seems to be more appropriate in experiment 2 than in experiment 1, as mentioned in the section above.

The KL-divergence score for experiment 2 are minimally lower than for experiment 1, suggesting the sense representations are slightly more suitable on average for all senses, and not just the nearest sense embedding. The significance of these differences is questionable – see chapter 6.

The clustering-approach in experiment 3 works relatively fine for the words ‘*ansigt', ‘plads'*, and again ‘*stand'*. There are quite many idiomatic expressions and figurative speeches for ‘*ansigt'*. Since the majority of data come from newswire text, these senses could be very common in the data, which enforce a consistency of sense features within the clusters.

Since the Korpus DK sentences are clustered around seeds from experiment 2, it could be suspected that the results generally follow the performance there, since this is their starting point. This is not evident from the results. Experiment 3 works rather non-satisfying for ‘*blik'*, ‘*lys'*, ‘*model'* and ‘*vold'* which in fact are outperformed by the random baseline. This suggests that the tuning of the seeds go in a wrong direction - the instances assigned to a cluster share other features than what is useful for WSD. This is the general case, except for ‘*ansigt'*, though the performance stays at around 0.28. The rather bad results from experiment 3 are also reflected in the KL-divergence scores. See discussion on this matter in chapter 6.

In summary, the words that generally are disambiguated most satisfactory are '*blik'*, '*hold'*, '*stand'*, '*top'* and '*selskab'*. All of these words have distinct information in the DanNet synsets, are homonyms or have non-subtle sense differences. Worst results are found for '*lys'* which has a high number of senses (16), but no huge advantage to the system since the inter-annotator agreement is relatively high (.81). Also, though the sense number is high, the senses are related in meaning and the differences are often very subtle. The word '*lys'* (and '*kort'*) is also a common adjective in Danish, which possibly affects the word embedding model. These reasons could explain why the system works less well for this word.

The size and intersections of the synsets seem to be important for the sense representations in experiment 1, where the sample sentence information for experiment 2 works best for homonyms.

Experiment 3 was motivated by the idea that sample sentences could work as seeds for clustering of more data, but both the accuracy and KL-divergence score (and machine learning classifiers, see more below and in Figure 19) proves otherwise, suggesting that the sample sentences do not have clear enough information as a base for clustering.

6.2 Experiment 4

Experiment 4 (classifiers on SemDaX and clustered data from exp. 3) shows that the classifiers perform best on within-dataset training and testing. The best performance is on the SemDaX data alone on 0.78 with a FFNN. The accuracy on the clustered data with a FFNN lies on 0.71. Both models significantly outperform the baseline (0.39 and 0.15, respectively), and the most frequent sense (0.56 and 0.18, respectively) too. This suggest that there is a clear pattern of features for the word senses in the data to be found by the FFNN algorithm. The picture is the same with the slightly less good SVM.

When mixing the data sets, the performance falls to 0.65, but still beats the baseline at 0.153 and the most frequent sense at 0.24. This suggests that there still is a pattern to find for the algorithm, but the function from features to the classes is more confused, which means that the features in one data set is a bit different from the features in the other data set for each class. This is not surprising, considering the origin of each data set. Again, this requires a systematic look in the differences between the data sets in future work.

This tendency continues when looking at the performance when the models are trained on the clustered data and tested externally on SemDax. The performance then falls to 0.18, which beat the baseline (0.147), and generally the (low) most frequent at 0.10.

The interesting results from experiment 4 are not those from the classifier trained on SemDaX exclusively, but the others. The motivation for this 4th experiment was to explore whether machine learning algorithms could find a function from features in data to classes (word senses). If the classifiers perform roughly the same within the SemDaX as on tested on the clustered data or a mix of the two sets, it could suggest that the features for each class are similar. If this was the case, it would seem plausible that the clustered data was clustered around appropriate seeds (considering the SemDaX as the objective). This was apparently not the case, even though the classifiers outperform the random baselines. For this reason, the features and classes in the clustered data seem to differ from those in SemDaX. The difference is not complete, but big enough to want to adjust the approach in the future.

In summary, all the classifiers in experiment 4 beat the baseline (random classification), but only the models tested (partly or exclusively) on the data they were trained on, perform better than the most frequent sense.

It is appropriate here once again to recall Pedersen et al (2018), who also trained a SVM on the target nouns in the SemDaX data. Here, the surrounding lemmas of the target words, and their order, were used as features. The model performs at around 0.66, on the same task, though only including 10 nouns. The SVM in this thesis work performs a bit higher on average on 9 of the 10 words, namely around 0.68. A cautious conclusion could be that representing the data instances as sentence centroids in the word embedded space trained on a huge corpus is a better feature than surrounding lemmas for word sense classification.

We have seen in the experiments that the parameters that influences the results are inter-annotator agreement, number of senses, synset sizes and synset member intersections, and the information in the sample sentences. Generally, and as expected, the words with most ambiguity are disambiguated poorest. The sense representation for '*Plade'* yields average performance, though the inter-annotator agreement for the word is very low, suggesting the sense representations here are not suitable. Shortcomings of the approach and possible biases of the results are now discussed below.

6.3 Linking from Dictionary Senses to DanNet Synsets

An obvious bias in the system is the necessary linking from dictionary sense labels in the evaluation data, to the DanNet-influenced built senses represented by a synset id. Several links from dictionary senses had to converge into one synset id, since there are no synsets for idiomatic expressions, which the evaluation data has been annotated with as well. On the one hand, the bias has been minimized by choosing the synset of the literal sense of the target word in the idiomatic expression, and would therefore hopefully affect the sense representation minimally. Also, the DanNet has been compiled from the Danish dictionary, which proposes a correspondence (there does exists key besides the manual created here, but is not available for research).
On the other hand, the labels before and after linking obviously cannot be said to be absolutely 1:1 (this will not be a problem in the future when accurate data is available), as DanNet is

coarser grained[19]. The bias will therefore be highest where several links converge, which is mostly where the frequency of idiomatic expressions are high. It would be possible to delete these instances from the data, but they make up quite an amount. Also, the expressions in its literate sense uses the target words as how they are considered when linked.
The *exact* quality of WSD of idiomatic expressions cannot be accounted for in this system, though.

Yet an aspect to consider is that not all the senses for each target word in DanNet have been included. This is the also the case, again, in the work of Pedersen et al. 2018, where they tested a SVM on sentences in SemDaX with the annotated senses. The senses not included in the system of this thesis work are therefore not in the evaluation data SemDaX, and could therefore be said to not be of big importance (un-used or rare) generally. This is of course not evident, and it might be so, that the non-included senses are very apparent in data elsewhere. This is yet to be investigated when future statistics on word sense distributions is covered.

6.4 WSD: Quality as Performance

The evaluation method used to test the word sense representation system is a WSD task. This assumes that if the created sense representations can disambiguate in a satisfying way, the system is of good quality. Though it is a customary evaluation method, it might not be true that the built sense representations are of bad quality, if it does not disambiguate test data well. Maybe it is the other way around: that the method to test the (possibly ideal) sense representations are not suitable. The test data can be refined and bigger to disqualify this concern, or another evaluation method can be applied (cluster-based or in an application).

Another point to discuss here in relation to the WSD task is the most frequent sense baseline, which the system is compared to. The most frequent annotation in the test data is chosen as the most frequent sense – but that might not be the case generally. Again, this is a matter of access to word sense frequencies in the Danish language in general. The 900 million word corpus that the word embeddings were trained on might have other sense frequencies than the test data. Even though the test data instances are represented in the vector space, we do not know which senses are the most frequent in the big training corpus, which probably would be the sense representation closest to the target word in the vector space. If a test data instance is very similar to the target word vector, the system will probably choose the most frequent sense (in the big corpus), but the point is, that we do not know whether that is the most frequent in the test data as well. Another normal baseline is the 1st sense (in the given lexical resource), which often is the most common sense. A random sense is also a possible alternative, which is also chosen in this present work.

[19] This is not only the case for DanNet, but is a general problem with wordnets, as these do not contain idiomatic expressions.

6.5 Data

The quality of data in any NLP system is crucial. Some discussion points on the data used in this thesis work are now given, as the data can be another major source of uncertainty.

The DanNet sample sentences are in experiment 3 supplemented with more data (sentences) using the k-means algorithm. Ideally, the new data should have no intersection with neither the DanNet sample sentences, the data from which the word embeddings were made nor the evaluation data SemDaX. We attempted to access such data, but due to practical reasons we decided on Korpus DK. It is uncertain whether there is an overlap, but probably there is.

Another possible imperfection arrives when using a lexical resource, as WordNet or the dictionary. Though these resources often are of good quality and fine-grained, they are rather static, descriptive and expensive as humans have made decisions on which elements to include, how, when and where etc. DanNet includes only a certain number of senses, has a certain granularity and structure, and treats rare word senses in the same way as common senses. Word embeddings are in contrast made automatically on the data given, and hence, possibly more up-to-date. The created sense representations in this work are anchored in wordnet associated data, and is therefore limited to the quality hereof. It is possible to add-on more data as in experiment 3, though the number of senses remain the same.

Word embeddings can also cause problems. Homonyms are treated as same element, and different inflections or class of a word are treated as different elements. It is also not possible to compare the word embeddings directly across vector spaces trained on different corpora. The homonym '*skade'* (harm/injury/damage, magpie, marine ray), used in this work, has several and non-related senses. As the word embedding model operates at word level, the word embedding for that word is affected by both senses at the same time, which is not necessarily desirable. This is not a big problem in the word sense representation system of this work though, as the sense representations are made from the centroids of the words in the sense-bag collection extracted from DanNet, and not induced from neighbour clusters to the target word embedding. Also, the hyperonymy relations are extracted, which skews the sense representations towards their proper "head" sense. The results of experiment 1 suggests that the hypernyms and hyponyms are valuable information to include in the system (see Table 6).

The annotation of the test data is also discussable. The disagreement is rather high, and only 2% of the data have been curated. Annotators report that the sentences often lack context and that the senses are rather fine-grained (Pedersen et al., 2018). As the senses can be very closely related, minor dissimilarities in how to understand the given word can influence the agreement. The disagreement can of course be lowered by different means, and will possibly decrease when the sense number drops, as the similarity between the possible senses might fall.

6.6 Manipulating Vectors and Averaging Information

All data used as input in the word sense representation system are word embeddings, that originally come from single words or sentences. Every sentence (either test data sentences, Korpus DK sentences or DanNet sentences) has been word tokenized, and represented as one vector in the word embedded space. Though this procedure is customary, it is a decision with consequences. There will inevitably be lost structural information of the sentence under data processing. However, some structure can be said to have lasted in the word embedding model, at least the co-occurrence information. This is a condition when modelling word similarity as in vector space models.

One decision is to treat a sentence as bag-of-words (possibly selected words). Another decision is how to transform the bag-of-words into one vector representation. In this thesis work, the mean of the word vectors is taken. Another possibility is to add the word vectors together. This will have consequences for the length of the resulting vectors, not the direction of them. As the cosine similarity is used in this thesis work as distance measure between vectors, the length of the vectors matters less, than if the Euclidean distance was used. Again, the task at hand can help identifying how vectors should be manipulated.

If certain words in a sentence are judged to have more importance than other words (e.g. those that play a semantic role to the target word), they could be extracted before representing them in a vector. Another possibility is to give the words a weight, which is applied when manipulating the vectors. This is indeed a possibility in future work.

6.7 The KL-divergence Metrics

It is obvious in Figure 4 that the two distributions that are compared with the KL-divergence score are rather dissimilar. On the one hand, it makes sense to treat each word sense (and annotation) as a valid candidate with assigned weights. As mentioned before, this is also the case and intuition in the SemEval 2013 task 13.

On the other hand, the method lead to high and very similar KL-divergence scores, whose differences cannot be said to be significant. In this work, each annotated sense is assigned the same probability. This does not properly rank all the word senses, but rather either give the word senses importance, or not. A possible solution could be to include statistics on how many annotators chose some senses over others, and give a higher weight to those senses that are chosen by, say, five of six annotators. This would be a more "democratic" approach, but would disregard the view that all annotators are just as right. A more expensive solution could be to adjust the annotator's task, and ask them to rank the possible word sense labels. These possible solutions could represent the evaluation data in a metric, which probably would be more similar to the metric of distances between the sense representations and a test sentence centroid. With two sets of sense labels, the system could be evaluated more similarly to the SemEval task 13 with a Jaccard Index and positionally-weighted Kendall's τ, and thereby use correlation statistics rather than divergence statistics.

6.8 Approach Assumptions

The approach of this thesis is to see whether sense representations based on wordnet associated data represented in a word embedded space, can disambiguate test data, which is also represented in the embedded space. The words in the sense-bags are grouped regardless of how they appear in the embedded space. This means that only a reasonable (distinctive) sense representation can be made if there is a correspondence with wordnet senses and how they actually behave in the big training corpus. The centroid of the words associated with each sense in DanNet can only be different from each other if the same groups of words also behave differently in the training data.

Also, the centroids of the test instance sentences can be affected by noise in the sentences. Some words are not important to the word sense. The context word with a window of 5 is considered, but the clues to which sense it is, might be elsewhere in the sentence.

The word sense representation system can be considered more or less guided. The sense embeddings made exclusively from words in DanNet (experiment 1 and 2) can be considered as new representations of the synsets, which are then applied for WSD. This is rather controlled and knowledge-based. Although, the sense representations are represented in the vector space model, which are rather data-driven. The clustering technique in experiment 3 can be said to be semi-supervised since new data unsupervised are assigned a class, which "happen" to have a label. Consequently, the system do not unsupervised induce and detect senses, but is guided. These aspects need to be considered in future comparison to unsupervised WSI systems.

6.9 Limitations and prospects

The approach of this thesis has a series of limitations, but also prospects.

A theoretical limitation is that the created sense representations are based on a lexical resource, which limits the created senses to the senses apparent in the given resource. This means there is a set number of senses, and the sense inventory would not cover other or new senses in data. Also, the system is limited by the quality of the given lexical resource. For example, some synsets in DanNet do not contain much information.[20] Unsupervised methods for word sense

[20] Practically, the system in this work is most likely more limited by the input data (with noise), and the approach limited by available evaluation data.

detection would be appropriate to detect new senses or senses of unknown words, but requires a suitable evaluation method.

Another limitation of the approach is that the machine learning classifiers can only be used to sense-tag the target words apparent in training data. A solution could be to auto-tag new training data using sense representations built like we do. Such a machine learning classifier could be trained for all words in the given lexical resource.

A practical limitation is the performance of the system. The sense representation system for experiment 1 to 3, which can be computed for any word in the given lexical ressource, does not outperform the most frequent sense, in fact, the best sense representations here only perform roughly half as good. Possible solutions have been discussed, and it is still an open question how the sense representations perform for less ambiguous words.

Nevertheless, the approach has some prospects. As mentioned above, it is possible to create sense representations for all senses apparent in the given lexical resource. Also, there is a number of possible rather straight-forward improvements (see next chapter), which could augment the quality of the sense embeddings. Finally, the supervised simple WSD classifiers from experiment 4 all outperformed the baselines significantly, and the best model (0.78) can be applied for DanNet sense-tagging of the lexical samples included in this work. This model also reaches the level of inter-annotator agreement in this sense inventory.

7 Conclusion

This study set out to determine the possibility of building appropriate sense representations for Danish sense tagging, by combining word embeddings with wordnet senses. The rationale is to combine corpus evidence with senses outlined by humans. Various data were extracted, represented in a word embedded space, and tested in a WSD task. Thousands of sample sentences were auto-tagged by sense clustering, and the clusters as well as the evaluation data were tested in machine learning classifiers. This technique to represent word sense could work, if the word senses behave approximately similar in both the (mostly) high-quality knowledge-based information and the distributional information of the given words.

As expected, wordnet-associated data is informative for the sense representations. Generally, the more semantic relations and information included from the wordnet, the better. The best WSD results for created sense representations are found in the first experiment, where the wordnet sample sentence, definition, local synset members and definition of the hypo- and hypernym are extracted all together. When comparing all annotations in all test instances to the sense embeddings through Kullback-Leibler divergence scores, the best sense representations across all target words come from the wordnet sample sentence alone. The differences in these scores are non-significant, and must be examined in future work.

Moreover, the word sense representation system in experiment 1-3 has other serious drawbacks. Firstly, it is generally far from performing as well as the most frequent sense as default, and it is not clear to what extent the mapping of dictionary senses in the evaluation data into wordnet synsets generates bias. Also, the machine learning classifiers works far better at the WSD task, but they are limited to the number of sense annotated words in data. Furthermore, the machine learning results indicate that the features of the auto-tagged clusters seems to have almost no connection at all to the sense features in the evaluation data.

However, the sense representation system can also be said to seem promising. The best sense embeddings in this pilot project do disambiguate better than by chance, even though the task is rather hard and the information not have been through (probably) valuable cleaning of data noise. With these aspects considered, the first pilot of this approach seems to function, though not yet suitable for application. Various improvements will possibly increase the quality of the representations leading to better disambiguations and possibly auto-tagged data. Especially for less polysemous words, which undoubtedly make up a bigger part of the Danish language than the selected highly ambiguous nouns included in this case study. Before evaluation of unsupervised WSI approaches is possible for Danish, it is now found that a combination of the lexical resource DanNet and word embeddings seems promising for future development of appropriate sense inventories for Danish language technologies.

Future work

The motivation for this thesis was partly to work towards comparing the supervised created clusters of word senses in Pedersen et al. 2018 (grouped by wordnet ontological types) to free-standing clusters induced unsupervised from data. Before curated open-source data is available to evaluate such clusters, it would be useful to explore how the senses represented as in this thesis work would cluster, and whether that would have a correspondence to the human-driven approach. Also, in line with the mentioned study, an exploration of adjustment of granularity of the sense inventory would be appropriate.

Another motivation for this thesis was to find word sense distributions, as these can e.g. improve sense-taggers or reveal qualities of different text types. The sense distributions could be found by ranking the sense embeddings by similarity to the target word, as Bhingardive et al. (2015) does for most frequent sense detection. For all words in DanNet, a sense embedding (and the nearest to the target word) can be found. These ideas are a start for a sense-tagger. Another possibility is to count tags in data tagged by a sense-tagger (such data can be evaluated e.g. in a survey or by a gold standard). The detections of the most common senses can also be compared to whether they are the 1st sense (in a lexical database), as this is usually the most common.

As sense representations from experiment 1 proved to work best compared to experiment 2, it would be suitable for near-future work to use these centroids as seeds in experiment 3. A clean-up of the sense-bags created from the sample sentences would also be an obvious next step: detection of the target word to extract the chosen context window would reduce noise in the data. The synonyms or the near-synonyms of words in the sample sentences could possibly augment the sense embeddings if these were included in the sense-bags. Also, when computing the centroids of the sense-bags, the neighbours could be weighted differently depending on the type of relation. Maybe a hypernym is more similar in the distributional sense than e.g. meronyms. The test sentences could also be clustered, and the centroid of those would be the sense embedding and seeds. This would limit a possible sense-tagger to the target nouns. Regarding experiment 4, some near-future work is also possible here. Firstly, the training data can be refined in line with the possible solutions given in this chapter. Secondly, the parameters set when training the classifiers can be tested more thoroughly which most likely will reveal some potential for better performance. Again, the classifiers can also be tested on less ambiguous nouns, if proper auto-tagged training data is created.

The sense representations can also be evaluated differently than in a WSD if suitable data are created. Such data can also benefit evaluation of un-supervised WSI systems where free-standing automatically induced clusters can be compared to. This can possibly reveal new senses and detect sense use over time. It can also discover sense distributions, i.e. by ranking the nearest cluster neighbours or count tags in auto-sense-tagged corpora, if the system is reliable. These possibilities open up for bootstrapping new data sets.

Another possible future project could be to adjust the sense embeddings by finding a relation (if such one exists) to the location of the test vectors. If the DanNet vectors are always a bit off (in a consistent way) to the gold standard sense vectors, the WSD guesses can be adjusted by knowledge on this relation. This would post-process a sense representation from the lexical resource into an adjusted sense representation, for instance by linear transformation.

Finally, a future expansion of this work could be to include other word classes, such as verbs, which also can be highly ambiguous. FrameNet will possibly be useful for this. Other work could be to test the approach on less polysemous words when data is available. At the time of writing, there is an ongoing project for evaluating the word embeddings. When these results are ready, it would be suitable to test this word sense representation system with the best word embedding model available. Lastly, as the approach of this thesis limits a possible sense-tagger to the words found in DanNet, a future project is to include unknown words.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *None*. https://doi.org/10.1038/nn.3331

Agirre, E., & Edmonds, P. (2006). Word Sense Disambiguation: Algorithms and Applications. *Text Speech and Language Technology*, *33*(33), 384. https://doi.org/10.1007/978-1-4020-4809-8

Agirre, E., & Soroa, A. (2007). Semeval-2007 Task 02 : Evaluating Word Sense Induction and Discrimination Systems. *Computational Linguistics*, (December), 7–12. Retrieved from http://www.aclweb.org/anthology/W/W07/W07-2002

Agirre, E., de Lacalle, O. L., & Soroa, A. (2014). Random Walks for Knowledge-Based Word Sense Disambiguation. *Computational Linguistics*. https://doi.org/10.1162/COLI_a_00164

Apidianaki, M. (2008). Translation-oriented Word Sense Induction Based on Parallel Corpora.

Asmussen, J. (2012). CLARIN-Referencekorpus. *Sprogteknologisk Workshop October 31*. Retrieved from http://cst.ku.dk/Workshop311012/sprogtekno2012.pdf

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley FrameNet Project. In *Proceedings of the 36th annual meeting on Association for Computational Linguistics -*. https://doi.org/10.3115/980845.980860

Barwise, J., & Cooper, R. (1981). Generalized Quantifiers and Natural Language. *Linguistics and Philosophy*, *4*(2), 159–219.

Basile, P., Caputo, A., & Semeraro, G. (2014). An Enhanced Lesk Word Sense Disambiguation Algorithm through a Distributional Semantic Model. In *the 25th International Conference on Computational Linguistics: Technical Papers (COLING'14)*. https://doi.org/10.1024/1012-5302/a000007

Bhingardive, S., Singh, D., & Murthy, R. (2015). Unsupervised Most Frequent Sense Detection using Word Embeddings. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1238–1243.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*. https://doi.org/10.1145/130385.130401

Bruce, R. F., & Wiebe, J. M. (1998). Word-sense distinguishability and inter-coder agreement. In *Proceedings of Third Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.3760/cma.j.issn.2095-4352.2015.07.008

Carnap, R. (1947). *Meaning and Necessity*. University of Chicago Press.

Chollet, F. (2015). Keras. *GitHub Repository*. Retrieved from https://github.com/keras-team

Chomsky, N. (1971). Deep structure, surface structure, and semantic interpretation. In *Semantics. An interdisciplinary reader in philosophy, linguistics and psychology*.

Clark, S. (2014). Vector space models of lexical meaning. *Handbook of Contemporary Semantics*, (March), 1–43. https://doi.org/10.1002/9781118882139.ch16

Cruse, D. A. (1986). *Lexical Semantics*. *Language*. https://doi.org/10.2307/414555

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B*. https://doi.org/10.2307/2984875

Denkowski, M. (2009). A Survey of Techniques for Unsupervised Word Sense Induction. *Language & Statistics II Literature Review*, 1–18. Retrieved from https://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/mdenkows/pdf/wsi2009.pdf

Dorow, B., & Widdows, D. (2003). Discovering Corpus-specific Word Senses. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 2* (pp. 79–82). Stroudsburg, PA, USA: Association for Computational Linguistics. https://doi.org/10.3115/1067737.1067753

Erk, K. (2012). Vector space models of word meaning, *10*(1), 1–24. https://doi.org/10.1002/lnco.362

Farley, B. G., & Clark, W. A. (1954). Simulation of self-organizing systems by digital computer. *IRE Professional Group on Information Theory*. https://doi.org/10.1109/TIT.1954.1057468

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database. MIT Press, Cambridge, London, England* (Vol. 71). https://doi.org/10.1139/h11-025

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

Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*. https://doi.org/10.1016/0004-3702(82)90020-0

Frege, G. (1892). Comments on sense and meaning. In *Posthumous writings*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. *Nature*.

https://doi.org/10.1038/nmeth.3707

Graf, P., & Schacter, D. L. (1985). Implicit and Explicit Memory for New Associations in Normal and Amnesic Subjects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*. https://doi.org/10.1037/0278-7393.11.3.501

Harris, Z. S. (1954). Distributional Structure. WORD, *10*(2–3), 146–162. https://doi.org/10.1080/00437956.1954.11659520

Hartigan, J. A., & Wong, M. A. (1979). A K-Means Clustering Algorithm. *Applied Statistics*. https://doi.org/10.2307/2346830

Hjorth, E., & Kristensen, K. (2005). *Den Danske Ordbog* (1st ed.). Copenhagen: Gyldendal.

Johansson, R., & Nieto Piña, L. (2015). Embedding a Semantic Network in a Word Space. *Naacl-2015*.

Iacobacci, I., Pilehvar, M. T., & Navigli, R. (2016). Embeddings for Word Sense Disambiguation: An Evaluation Study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. https://doi.org/10.18653/v1/P16-1085

Jones, E., Oliphant, T., Peterson, P., & others. (2001). SciPy: Open source scientific tools for Python. Retrieved from http://www.scipy.org/

Jurafsky, D., & Martin, J. H. (2009). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. *Speech and Language Processing An Introduction to Natural Language Processing Computational Linguistics and Speech Recognition*. https://doi.org/10.1162/089120100750105975

Jurgens, D., & Klapaftis, I. (2013). SemEval-2013 Task 13: Word Sense Induction for Graded and Non-Graded Senses. *Seventh International Workshop on Semantic Evaluation*.

Kilgarriff, A. (1997). “I don’t believe in word senses.” *Computers and the Humanities*. https://doi.org/10.1023/A:1000583911091

Kilgarriff, A. (1999). 95% Replicability for Manual Word Sense Tagging. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics* (pp. 277–278). Bergen, Norway: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/977035.977084

Kilgarriff, A., & Palmer, M. (2000). Introduction to the special issue on SENSEVAL. *Computers and the Humanities*. https://doi.org/10.1023/A:1002619001915

Kilgarriff, A. (2001). English lexical sample task description. *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, 17–20.

Klapaftis, I. P., & Manandhar, S. (2008). Word Sense Induction Using Graphs of Collocations. *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*. https://doi.org/10.3233/978-1-58603-891-5-298

Kripke, S. (1980). *Naming and Necessity*. Harvard University Press.

Kågebäck, M., Johansson, F., Johansson, R., & Dubhashi, D. (2015). Neural context embeddings for automatic discovery of word senses. *Workshop on Vector Modeling for NLP*, 25–32.

Kågebäck, M., & Salomonsson, H. (2016). Word Sense Disambiguation using a Bidirectional LSTM. *5th Workshop on Cognitive Aspects of the Lexicon (CogALex)*.

Lakoff, G. (1971). On Generative Semantics. In *Semantics: An Interdisciplinary Reader*. https://doi.org/10.1016/j.socscimed.2013.05.029

Landauer, T. K., Folt, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*. https://doi.org/10.1080/01638539809545028

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries. In *Proceedings of the 5th annual international conference on Systems documentation - SIGDOC '86*. https://doi.org/10.1145/318723.318728

Lloyd, S. P. (1982). Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*. https://doi.org/10.1109/TIT.1982.1056489

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. *Journal of Endodontics*. https://doi.org/10.1097/00004770-200204000-00018

Lowe, W. (2001). Towards a Theory of Semantic Space. *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*.

MacQueen, J. B. (1967). Kmeans Some Methods for classification and Analysis of Multivariate Observations. *5th Berkeley Symposium on Mathematical Statistics and Probability 1967*. https://doi.org/citeulike-article-id:6083430

Melamud, O., Goldberger, J., & Dagan, I. (2016). context2vec: Learning Generic Context Embedding with Bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. https://doi.org/10.18653/v1/K16-1006

Meyer, D., & Schvaneveldt, R. (1971). Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology*, *90*, 227–234.

Mihalcea, R., Chklovski, T., & Kilgarriff, A. (2004). The SENSEVAL-3 English Lexical Sample Task. *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL-3)*. https://doi.org/10.1027/1016-9040.10.1.79

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space, 1–12. https://doi.org/10.1162/153244303322533223

Montague, R. (1970). Universal grammar. *Theoria*. https://doi.org/10.1111/j.1755-2567.1970.tb00434.x

Moro, A., Raganato, A., & Navigli, R. (2014). Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics (TACL)*.

Palmer, M., Dang, H. T., & Fellbaum, C. (2007). Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*. https://doi.org/10.1017/S135132490500402X

Pantel, P., & Lin, D. (2002). Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*. https://doi.org/10.1145/775047.775138

Pedersen, B. S., Agirrezabal, M., Nimb, S., Olsen, S., & Rørmann, I. (2018). Towards a principled approach to sense clustering – a case study of wordnet and dictionary senses in Danish. *Proceedings of Global WordNet Conference*, 1–8. Retrieved from http://compling.hss.ntu.edu.sg/events/2018-gwc/pdfs/GWC2018_paper_27.pdf

Pedersen, B. S., Braasch, A., Johannsen, A. T., Martinez Alonso, H., Nimb, S., Olsen, S., … Sørensen, N. (2016). The SemDaX Corpus - sense annotations with scalable sense inventories. In *Proceedings of the 10th conference of the Language Resources and Evaluation Conference* (pp. 842–847). European Language Resources Association.

Pedersen, B. S., Nimb, S., Asmussen, J., Sørensen, N. H., 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. https://doi.org/10.1007/s10579-009-9092-1

Pedregosa, F., & Varoquaux, G. (2011). *Scikit-learn: Machine learning in Python. ... of Machine Learning ...*. https://doi.org/10.1007/s13398-014-0173-7.2

Pelevina, M., Arefyev, N., Biemann, C., & Panchenko, A. (2017). Making Sense of Word Embeddings, (2012). Retrieved from http://arxiv.org/abs/1708.03390

Pustejovsky, J. (1995). *The Generative Lexicon (Language, Speech, and Communication*. Cambridge, Massachusetts: The MIT Press.

Pustejovsky, J., & Jezek, E. (2016). Integrating Generative Lexicon and Lexical Semantic Resources. *LREC 2016 Tutorial*. Retrieved from http://lrec2016.lrec-conf.org/media/filer_public/2016/05/10/tutorialmaterial_pustejovsky.pdf

Rehurek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP*

Frameworks (pp. 45–50). Valletta, Malta: ELRA.

Rochester, N., Holland, J. H., Haibt, L. H., & Duda, W. L. (1956). Tests on a cell assembly theory of the action of the brain, using a large digital computer. *IRE Transactions on Information Theory*. https://doi.org/10.1109/TIT.1956.1056810

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*. https://doi.org/10.1037/h0042519

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*. https://doi.org/10.1038/323533a0

Salton, G. (1968). Information Storage and Retrieval. Reports on Analysis, Search, and Iterative Retrieval. *Report: ISR-14. 456pp. Oct 1968*.

Schütze, H. (1998). Automatic Word Sense Discrimination. *Computational Linguistics*.

Shen, H., Bunescu, R., & Mihalcea, R. (2013). Coarse to Fine Grained Sense Disambiguation in Wikipedia. In **SEM 2013 - 2nd Joint Conference on Lexical and Computational Semantics*. https://doi.org/10.1002/mus.21274

Song, L., Wang, Z., Mi, H., & Gildea, D. (2016). Sense Embedding Learning for Word Sense Induction, (1), 85–90. https://doi.org/10.18653/v1/S16-2009

Sørensen, N. H., & Nimb, S. (2018). Word2Dict - Lemma Selection and Dictionary Editing Assisted by Word Embeddings. *Proceedings of the 18th EURALEX International Congres: Lexocography in Global Contexts*, 819–827. https://doi.org/10.4312/9789610600961

Steedman, M. (2000). *The Syntactic Process*. *Computational Linguistics*.

Tang, G., Rao, G., Yu, D., & Xun, E. (2016). Can We Neglect Function Words in Word Embedding?
van Rossum, G. (1995). *Python tutorial*. Amsterdam.

Vapnik, V., & Lerner, A. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control*. https://doi.org/citeulike-article-id:619639

Yuan, D., Richardson, J., Doherty, R., Evans, C., & Altendorf, E. (2016). Semi-supervised Word Sense Disambiguation with Neural Models. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

Wittgenstein, L. (1921). *Tractatus Logico Philosophicus*. *Tractatus Logico Philosophicus Routledge Classics Routledge Classics*.

Zhong, Z., & Ng, H. T. (2010). It Makes Sense: A Wide-coverage Word Sense Disambiguation System for Free Text. In *Proceedings of the 48th ACL*.

Google Code Archive, word2vec project: "Google Code Archive - Long-term storage for Google Code Project Hosting". *code.google.com*. Retrieved 2018-06-13