Best of Both Worlds: Making Word Sense Embeddings Interpretable

Alexander Panchenko
Language Technology Group, Technische Universität Darmstadt
Hochschulstr. 10, 64289, Darmstadt, Germany
panchenko@lt.informatik.tu-darmstadt.de

Abstract

Word sense embeddings represent a word sense as a low-dimensional numeric vector. While this representation is potentially useful for NLP applications, its interpretability is inherently limited. We propose a simple technique that improves interpretability of sense vectors by mapping them to synsets of a lexical resource. Our experiments with AdaGram sense embeddings and BabelNet synsets show that it is possible to retrieve synsets that correspond to automatically learned sense vectors with Precision of 0.87, Recall of 0.42 and AUC of 0.78.

Keywords: word sense embeddings, WordNet, BabelNet, AdaGram, sense matching, lexical semantics

1. Introduction

Two main approaches to represent the meaning of lexical units, such as words and multiword expressions are lexicography and statistical corpus analysis. In the first approach, a human explicitly encodes lexical-semantic knowledge, usually in the form of synsets (i.e. sets of synonyms), typed relations between synsets and sense definitions. A prominent example of this approach is the Princeton WordNet (Miller, 1995). The second approach makes use of text corpora to extract relations between words and feature representations of words and senses. These methods are trying to avoid manual work as much as possible. Whereas lexical resources are manually created, in the second approach most methods extract the information from text without human intervention. Examples of the second group of methods include "classical" vector-based (Baroni and Lenci, 2010) and symbolic (Biemann and Riedl, 2013) distributional models, as well as word embeddings (Mikolov et al., 2013; Pennington et al., 2014).

One of the strongest sides of lexical-semantic resources is their *interpretability* – they are entirely human-readable and drawn distinctions are motivated by lexicographic or psychological considerations. On the downside, these WordNet-like resources are expensive to create, and it is not easy to adapt them to a given domain of interest or language. Besides, sense inventories of lexical resources are often too fine grained to be useful in downstream applications (Brown, 2008).

At the same time, corpus-driven approaches are strong at *adaptivity* – they can be re-trained on a new corpus, thus naturally adapting to the domain at hand. If fitted with a word sense induction algorithm, corpus-driven approaches can also discover new senses (Erk et al., 2009). However, the representations they deliver are often not matching the standards of lexicography, and they rather distinguish word usages than senses. Moreover, dense numeric vector representations as present in latent vector spaces (Schütze, 1998) and word embeddings are barely interpretable.

Word sense embeddings (Huang et al., 2012; Tian et al., 2014; Neelakantan et al., 2014) extend word embeddings so that a word is represented by several vectors corresponding to meanings of the word. Li and Jurafsky (2015) show that sense embeddings can significantly improve performance of part-of-speech tagging, semantic relation identification and semantic relatedness tasks, but yield no improvement for named entity recognition and sentiment analysis. Sense embeddings suffer the same interpretability limitations as other dense vector representations.

The contribution of the paper is a technique that links word sense embeddings to a lexical resource, making them more interpretable. The main motivation of the technique is to close the gap between interpretability and adaptivity of lexical-semantic models. We demonstrate the performance of our method by linking AdaGram sense embeddings, proposed by Bartunov et al. (2015) to synsets of BabelNet (Navigli and Ponzetto, 2010). However, the approach can be straightforwardly applied to any combination of a WordNet-like resource and a word sense embeddings model. Scripts and datasets related to this experiment are available online.[1]

To our knowledge, this is the first attempt to tag sense embeddings with interpretable synsets from a lexical resource. While other approaches exist that use distributional information for enriching lexical resources (c.f. the next section), we are not aware of any other approach that utilizes corpus-induced senses in the form of sense embeddings for this purpose.

2. Related Work

Aligning senses across several lexicographic resources has been sought as a means to achieve more comprehensive sense inventories. Recent approaches include methods used to build BabelNet and UBY (Gurevych et al., 2012). Both of these lexical resources automatically interlink word senses across multiple dictionaries and encyclopaedias, such as Wiktionary[2], Wikipedia[3] and Omega Wiki[4]. This line of research is focused on interlinking manually created lexical resources. However, they not attempt to align any corpus-driven sense inventory.

While sense coverage and disambiguation coverage is increased through more and richer sense representations,

[1]http://tudarmstadt-lt.github.io/vec2synset
[2]http://www.wiktionary.org
[3]http://www.wikipedia.org
[4]http://www.omegawiki.org

these extended resources suffer from alignment errors, as well as the disadvantages of lexicographic resources as discussed in the introduction.

While lexicographic work mostly relies on corpus-based, yet hand-picked evidence, Hanks (2013) presents an approach to systematize and formalize this approach, based on word characteristics as yielded by the Sketch Engine corpus analysis tool (Kilgarriff et al., 2014).

The need of corpus-based adaptation of lexical resources is discussed by McCarthy et al. (2004), who define a method to find the dominant sense of a word with respect to a text collection, in order to inform the most frequent sense baseline in word sense disambiguation. In (Agirre et al., 2006), automatically induced senses are mapped to WordNet via hand-labelled instances in the training set.

Automatically induced sense inventories were used in word sense disambiguation tasks by Biemann (2010), yet as features and without explicit mapping to WordNet senses.

While most word embedding approaches represent a term with a single vector and thus conflate senses, there are few approaches to produce word sense embeddings from corpora (Huang et al., 2012; Tian et al., 2014; Neelakantan et al., 2014; Bartunov et al., 2015; Li and Jurafsky, 2015). However, these representations have, to our knowledge, not been directly mapped to a lexicographic resource.

Approaches that compute embeddings directly on knowledge bases are presented by Bordes et al. (2011) and Camacho-Collados et al. (2015). Rothe and Schütze (2015) combine un-disambiguated embeddings to WordNet synset to obtain synset representations in the embeddings space. The approach is evaluated on lexical sample tasks by adding synset embeddings as features to an existing WSD system. While this setup is flexible with respect to the kinds of embeddings used, it requires a large number of training instances per lexeme and is not able to find new senses in the underlying corpora. Our approach is different as we do not try to learn embeddings for all synsets in a lexical resource, but instead retrieve synsets that correspond to input sense embeddings.

3. Two Worlds of Lexical Semantics

This sections describes the two resources we link with our method and their comparison.

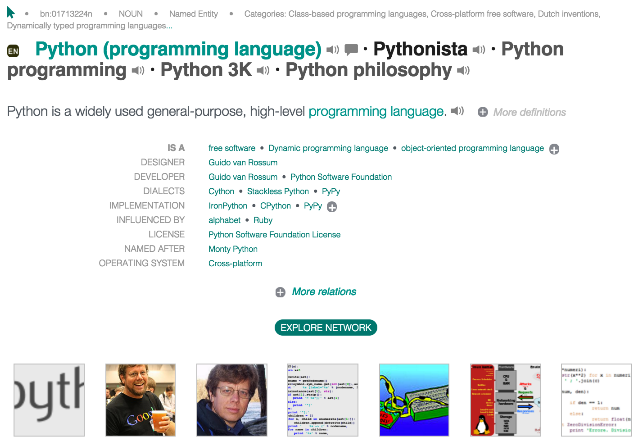
3.1. Lexicographic Resource: BabelNet

BabelNet consists of several lexical, such as WordNet, and crowd-constructed resources, such as Wikipedia, Wiktionary and Freebase[5], which are aligned semi-automatically across different languages. For our experiments, we use the English part of BabelNet in version 3.0.

BabelNet represents a word sense with a synset consisting of a set of lexical items, definitions and taxonomic relations. BabelNet synsets are easily interpretable as they feature explicit sense definitions, manually selected usage examples complemented by additional features that help to grasp word meaning, such as pictures illustrating the sense, taxonomic relations and even domain information as illustrated on Figure 1.[6]

[5]http://www.freebase.org

[6]http://babelnet.org/synset?word=bn:01713224n


• bn:01713224n • NOUN • Named Entity • Categories: Class-based programming languages, Cross-platform free software, Dutch inventions, Dynamically typed programming languages...

EN **Python (programming language)** • **Pythonista** • **Python programming** • **Python 3K** • **Python philosophy**

Python is a widely used general-purpose, high-level programming language. More definitions

IS A	free software • Dynamic programming language • object-oriented programming language
DESIGNER	Guido van Rossum
DEVELOPER	Guido van Rossum • Python Software Foundation
DIALECTS	Cython • Stackless Python • PyPy
IMPLEMENTATION	IronPython • CPython • PyPy
INFLUENCED BY	alphabet • Ruby
LICENSE	Python Software Foundation License
NAMED AFTER	Monty Python
OPERATING SYSTEM	Cross-platform

More relations

EXPLORE NETWORK

Figure 1: BabelNet synset `bn:01713224n` that corresponds to the word "python" in the programming language sense. Definitions, synonyms, taxonomic relations and images make this representation easily interpretable.

3.2. Word Sense Embeddings: AdaGram

The training objective of the Skip-gram model (Mikolov et al., 2013) is to find vector word representations that are useful for predicting the surrounding words in a sentence or document. The model represents a word sense as a low-dimensional vector. AdaGram (Bartunov et al., 2015) is a Bayesian nonparametric extension of the Skip-gram model that learns several embeddings per word corresponding to word senses.

We chosen AdaGram for experiments as they outperform the approach of Neelakantan et al. (2014) according to several word sense disambiguation benchmarks and have an open source implementation in contrast to the method of Li and Jurafsky (2015). Few other approaches have important limitations, so we did not consider them either. For instance, the method of Tian et al. (2014) assumes a fixed number of senses for all words, which is undesirable due to exponential distribution of number of senses. The approach of Huang et al. (2012) performs offline clustering of word contexts and thus is computationally expensive for large corpora. On the other hand, AdaGram can be considered as an online clustering of contexts, which therefore can scale to large corpora keeping a reasonable memory footprint. It is scalable due to an online variational learning algorithm used for training. As opposed to (Tian et al., 2014), the number of prototypes is found automatically, while senses granularity is regulated by the resolution parameter α.

As opposed to the interpretable BabelNet representation illustrated in Figure 1, an embedding is a dense vector typically in a 100-500 dimensional space, where the meaning of the dimensions is not specified in any human-interpretable format. The vectors themselves are therefore uninterpretable by humans.

However, one can interpret a sense vector by comparing it to other vectors. Namely, a list of nearest neighbours in the vector space can be used to interpret a sense. For instance, ten nearest neighbours of the vector corresponding to the word "python" in the programming language sense obtained during our experiments is as following: "perl, php, java, smalltalk, ruby, lua, tcl, scripting, javascript, bindings". While in some cases, this list of related words can be sufficient for interpretation of a word sense, it does not contains the wealth of additional information present in BabelNet synsets, such as taxonomic relations, human-readable definitions, images, and so on.

For the purpose of this paper, we have trained[7] an AdaGram model on an English corpus consisting of Web documents and Wikipedia articles with the default meta-parameters. Namely the resolution parameter α was set to 0.05, yielding 4.2 senses per word in average, and the number of dimensions in sense vectors was set to 100. The choice of the default parameters is dictated by the goal of our experiment. The idea was to show feasibility of linking two resources, rather than finding an optimal granularity for such alignment. In particular, we used surface forms of ukWaC and WaCkypedia_EN corpora by Baroni et al. (2009).[8]

3.3. Comparison of AdaGram and BabelNet Word Sense Inventories

There are often considerably more BabelNet senses than AdaGram senses. From Figure 2, we can observe a huge discrepancy in granularity of their sense inventories. The inventory of the embeddings is coarse-grained, while inventory of the lexical resource is extremely fine-grained. The maximum number of senses in our AdaGram model, controlled by α parameter, is five, while for the BabelNet it reaches up to 200 senses for some words.

Inspection revealed that many of these senses are named entities, such as three roller-coasters named "Python" with BabelNet IDs `bn:00279773n`, `bn:03501078n` and `bn:14645852n`. Some words, like "pilot" "angel" or "gold", have over 100 senses in BabelNet, many representing rare named entities.

Our technique can be used to tag any word sense embeddings with synsets from any WordNet-like resource, but BabelNet is especially well suited in this context due to its high coverage of different domains, rare senses and multiple languages. The more senses lexical resource contains the higher the probability that an automatically induced sense will be linked to a synset.

4. Linking Embeddings to Synsets

Our matching technique takes as input a trained word sense embeddings model, a set of synsets from a lexical resource and outputs a mapping from sense embeddings to synsets of the lexical resource. The method includes four steps. First, we convert word sense embeddings to a lexicalized representation and perform alignment via word overlap. Second we build a bag-of-word (BoW) representation of synsets. Third, we build a bag-of-word representation of sense embeddings. Finally, we measure similarity of senses and link most similar vector-synset pairs. Below we present each step in detail.

[7]https://github.com/sbos/AdaGram.jl

[8]http://wacky.sslmit.unibo.it/doku.php?id=corpora

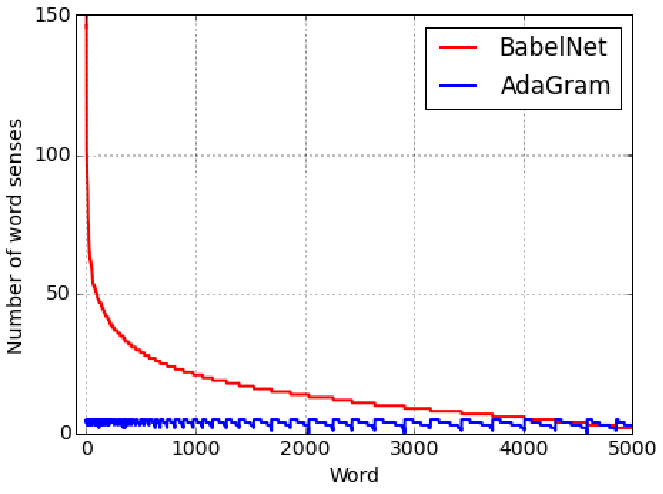


Figure 2: Comparative number of senses in BabelNet and AdaGram for 4724 frequent words.

4.1. Representation of Synsets

A bag-of-words that represents a synset is constructed of synset words, glosses, categories assigned to the synset and captions of images representing the synset. Glosses, categories and captions are lemmatized and cleaned from stopwords. For morphological analysis in our experiment we rely on SpaCy.[9] A word in a bag-of-words is weighted with its normalized frequency $w \in [0;1]$: We build frequency dictionary of words coming from synsets, glosses, categories and captions and simply normalize word counts by the largest word count.

4.2. Representation of Sense Embeddings

For each sense vector, we build a bag-of-words featuring the 200 most similar words and their lemmas according to the AdaGram model using the built-in function for nearest neighbors computation. Similarity of a sense to its neighbours is used as bag-of-word weight $w \in [0;1]$.

4.3. Linking Sense Representations

We experimented with two strategies that link sense vectors to synsets: the *global threshold* and the *disambiguation*.

4.3.1. Linking via Global Threshold

Let a word have n synsets and m sense vectors each represented by a bag-of-words. Then, in the first approach, we would calculate all $n*m$ pairwise similarities between these senses and link pairs with similarity above certain *global threshold* t, where t is the same for all words:

$$match(v_i, s_j) = \begin{cases} 1 & \text{if } sim(v_i, s_j) \geq t \\ 0 & \text{otherwise} \end{cases}$$

Here $sim(v_i, s_j)$ is a similarity score between a bag-of-word representing a sense vector v_i and a bag-of-word representing the synset s_j. The similarity between bag-of-words is calculated with either *cosine* or *overlap*. The global threshold method enables many-to-many mapping desirable in this context. As exemplified in Table 1, the

[9]https://spacy.io

Word	AdaGram ID	BabelNet ID	Sim.	AdaGram BoW	BabelNet BoW
python	2	bn:01713224n	0.103	perl, php, java, smalltalk, ruby, lua, tcl, scripting, javascript, bindings, binding, programming, coldfusion, actionscript, net, . . .	language, programming, pythonista, python programming, python3, python2, level, computer, pythonistas, python3000, python, . . .
python	1	bn:01157670n	0.102	monty, circus, spamalot, python, magoo, muppet, snoopy, featurette, disney, tunes, tune, classic, shorts, short, apocalypse, . . .	monty, comedy, monty python, british, monte, monte python, troupe, pythonesque, foot, artist, record, surreal, terry, . . .
python	3	bn:00046456n	0.066	spectacled, unicornis, snake, giant, caiman, leopard, squirrel, crocodile, horned, cat, mole, elephant, opossum, pheasant, zebra, . . .	molurus, indian, boa, tigris, tiger python, rock, tiger, indian python, reptile, python molurus, indian rock python, coluber, bivittatus, . . .
python	4	bn:01157670n	0.063	circus, fly, flying, dusk, lizard, moth, unicorn, puff, adder, vulture, tyrannosaurus, zephyr, badger, . . .	monty, comedy, monty python, british, monte, monte python, troupe, pythonesque, foot, artist, record, surreal, terry, . . .
python	1	bn:00473212n	0.060	monty, circus, spamalot, python, magoo, muppet, snoopy, featurette, disney, tunes, tune, classic, shorts, short, apocalypse, . . .	pictures, monty, python monty pictures, limited, company, python pictures limited, kingdom, picture, serve, director, united, five, . . .
python	1	bn:03489893n	0.056	monty, circus, spamalot, python, magoo, muppet, snoopy, featurette, disney, tunes, tune, classic, shorts, short, apocalypse, . . .	film, horror, movie, clabaugh, richard, monster, century, direct, snake, python movie, television, giant, natural, language, for-tv, . . .

Table 1: Result of the mapping of the AdaGram sense embeddings to the BabelNet synsets for the word “python” with the threshold of 0.05. The AdaGram BoW contains top nearest neighbours in the vectors space, while the BabelNet BoW contains most frequent words from synset, related words and glosses. This mapping helps to interpret sense vectors linking them to human-understandable synsets available by the BabelNet ID (c.f. Figure 1).

“Monty Python” sense of the word “python” is represented with two sense embeddings (AdaGram IDs 1 and 4) and two synsets (BabelNet IDs `bn:01157670n` and `bn:00473212n`).
Sample output of the mapping between sense embeddings and synsets of the word “python” is presented in Table 1. Further examples of linking AdaGram embeddings to BabelNet of 4724 frequent words are available online.[10]

4.3.2. Linking via Disambiguation

The second linking approach starts with *disambiguation* of a synset using the corresponding built-in AdaGram function, which performs a Bayesian inference based on the learned model, c.f. (Bartunov et al., 2015). Namely, we pass a list of words from the bag-of-words to this function as context of the target word. Next, we decide to link the assigned sense depending on similarity of either *confidence* of disambiguation or *overlap* of the bag-of-words of v_i and s_j. Here again we rely on the global threshold t of these similarities, but in the second strategy, one vector is linked to at most one synset.

5. Evaluation

We evaluate our linking techniques with respect to a manual mapping of senses. In particular, we built an evaluation dataset for 50 ambiguous words presented in Table 2. More specifically, we selected words with homonymous senses i.e. senses with unrelated meanings, such as “python” in the animal and the programming language senses. Some of these words, such as “bank” and “plant” are commonly used in word sense disambiguation evaluations (Navigli et al., 2007; Manandhar et al., 2010; Jurgens and Klapaftis, 2013); others, like “delphi” or “python” may refer to both nouns and named entities.

For each of these words, we retrieved all BabelNet and AdaGram senses. Next, we generated all 3795 possible matching combinations for these 50 words and annotated them binarily.

As mentioned above, BabelNet is very fine grained and contains more senses than AdaGram. Word sense embeddings used in our experiments are on the countrary coarse-grained with at most five senses per word (this is tunable by the α parameter). Therefore, the corpus-based model cannot learn a fine-grained polysemic inventory featuring tens of senses. For instance, AdaGram does not contain separate senses for “apple fruit”[11] and “apple fruit as a symbol”[12] found in BabelNet. Instead, it contains senses that correspond to “apple (computer)” and “apple (fruit)”. That is why, during annotation, we often positively linked a coarse-grained AdaGram sense to a more specific BabelNet sense. The negatively linked senses are those with completely unrelated meanings, such as “apple” in the company and fruit senses. The final dataset contains 423 positive and 3372 negative sense alignments.[13]

[10]Result of linking of AdaGram sense embeddings to BabelNet synsets for 4724 frequent words: https://goo.gl/dN6WSG

[11]http://babelnet.org/synset?word=bn:00005054n
[12]http://babelnet.org/synset?word=bn:00319426n
[13]Evaluation dataset: https://goo.gl/F2kuBA

ant (3|11), apache (3|19), apollo (3|28), apple (4|15), atom (2|19), bank (4|29), bass (5|19), blizzard (2|9), bosch (3|5), brother (4|24), canon (5|18), capital (4|28), cassandra (2|20), citizen (4|7), cloud (4|24), cobra (3|34), commercial (5|10), corvette (2|5), delphi (2|10), focus (5|38), jaguar (4|21), java (4|17), jena (2|8), julia (4|30), lotus (5|23), market (3|13), mouse (5|22), mustang (4|13), network (4|19), oracle (4|25), organ (5|11), pascal (3|10), plant (5|17), port (4|25), puma (3|19), python (4|17), raspberry (3|8), ruby (4|39), rust (4|17), sex (5|25), shell (5|33), soul (4|18), spark (4|37), sphinx (2|24), spider (5|24), tiger (4|35), tomcat (2|7), viper (3|24), vladimir (3|11), word (5|17)

Table 2: List of ambiguous words used in the evaluation dataset. Here "ant (3|11)" denotes that the word "ant" has 3 AdaGram senses and 11 BabelNet senses. Each word has at least two homonymous senses, e.g. the word "ant" can denote an insect sense, but also the Java build tool "Apache Ant".

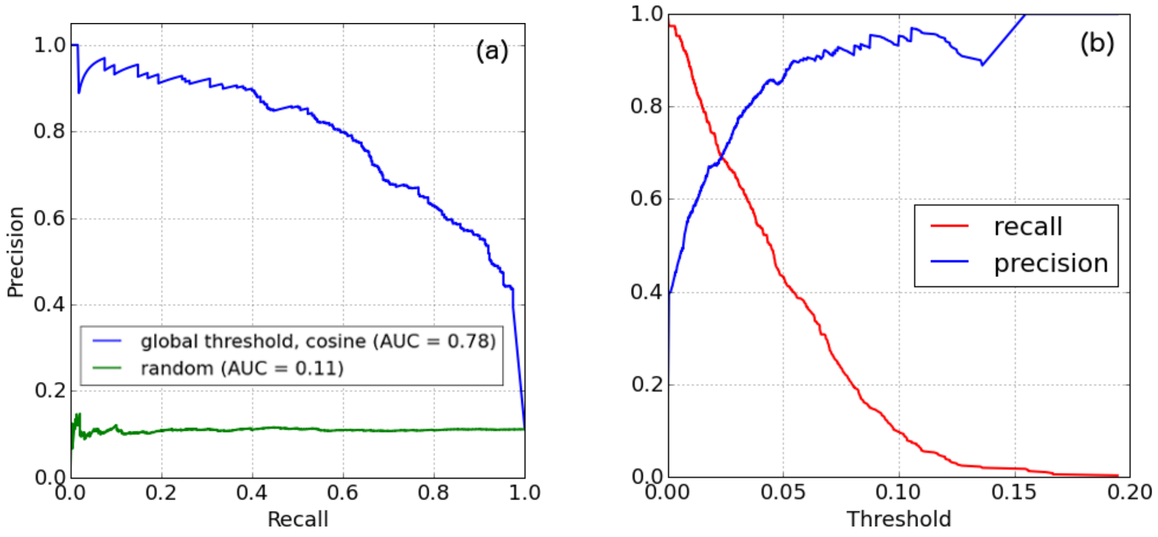


Figure 3: (a) Precision-recall curve of the best matching method *global threshold cosine* compared to the random mapping. (b) Precision and recall of the the same method function of the threshold t.

Method	BoW Similarity	AUC
random	random	0.11
disambiguation	confidence	0.53
disambiguation	overlap	0.66
global threshold	overlap	0.73
global threshold	cosine	**0.78**

Table 3: Performance of methods for linking AdaGram sense embeddings to BabelNet synsets on the evaluation dataset of 50 ambiguous words.

Class	Precision	Recall	F-measure	Support
match	0.87	0.42	0.57	423
no match	0.93	0.99	0.96	3372

Table 4: Performance of the best linking method *global threshold cosine* on the evaluation dataset of 50 ambiguous words at the threshold value of 0.05.

6. Results

Table 3 presents key results of our experiments. The baseline that assigns a random synset to a sense vector has an area under precision-recall curve (AUC) of 0.11.

The matching by built-in AdaGram *disambiguation* sounds attractive, as it relies not only on word overlap, but also on word similarities encoded in the embeddings. Yet, we observe that it fails for BabelNet senses that have no correspondence in the corpus-induced senses. AdaGram always assigns one of the senses from its inventory, and in such cases provides no meaningful indication of disambiguation confidence. The comparably low AUC of 0.53 of the *disambiguation, confidence* method based on confidence scores of the AdaGram, shows that these scores cannot be used to robustly rank pairs of candidate senses. Using the word overlap instead confidence of disambiguation increases AUC from 0.53 to 0.66.

According to our experiments, the best way to map senses is simply to calculate cosine between their bag-of-words and then link vector-synset pairs with the similarity above certain threshold. This approach yields an AUC of 0.78 (see also Figure 3 (a)). Ranking of sense pairs by overlap provides slightly worse results with AUC of 0.77, showing the utility of similarity scores in this task.

Figure 3 (b) depicts dependence of the precision and recall from the threshold value t. Precision increases with the value of the threshold and thus a user may select the value that fits best her use-case. For instance, a t value of 0.05 corresponds to precision of 0.87 and recall of 0.42. Table 4 provides a breakdown of the precision and recall scores at the threshold value t of 0.05 for the *global threshold* method using the cosine similarity.

A relatively low recall of 0.42 at the precision level of 0.87 is caused by two issues. First, some BabelNet synsets have

void bag-of-words as no text is associated with their English synset. For instance, the sense `bn:14200967n` of the word "delphi" and the sense `bn:14944150n` of the word "java" have no English definitions (this problem may be fixed in future releases). Second, some positive vector-synset pairs are not linked as similarity of their BoW representations is below the threshold t of 0.05.
For instance, the BabelNet sense of the word "apple" represented with the bag-of-words "store, retail, apple store, stores, chain, apple retail store, applestore, boston,... " has a cosine similarity of 0.002 with the AdaGram vector represented with the BoW "macintosh, hardware, pc, microsoft, ibm, pcs, dos, emulator, computers, os, beos,...". Yet, both these senses are strongly related. Therefore, a major problem with recall is caused by sparseness of the bag-of-words representation. Prior research (Bär et al., 2012) suggest that this limitation can be addressed using word relatedness measures.

7. Conclusion

Interpretation of clustering results is an inherently difficult problem. Word sense induction via sense embeddings is based on clustering of word contexts and therefore also faces this challenge. We propose a simple yet effective technique that improves interpretability of embeddings by linking them to highly human-readable synsets of a lexical resource, featuring proper definitions, examples of usage, and so on. The approach is able to link up to 42% of induced senses with precision of 87%. In addition to interpretability, our approach gives rise to hybrid methods (e.g. word sense disambiguation) that rely on information from both corpora and lexical resources.

8. Acknowledgements

This research was supported by the Deutsche Forschungsgemeinschaft under the project "Joining Ontologies and Semantics Induced from Text" (JOIN-T). Thorough comments of Chris Biemann, Martin Riedl, Benjamin Milde and three anonymous reviewers helped to significantly improve quality of this paper. Besides, I thank Sergey Bartunov for making implementation of the AdaGram available and for explaining how to operate the system.

9. Bibliographical References

Agirre, E., Martínez, D., López de Lacalle, O., and Soroa, A. (2006). Evaluating and optimizing the parameters of an unsupervised graph-based wsd algorithm. In *Proceedings of TextGraphs*, pages 89–96, NY, USA.

Bär, D., Biemann, C., Gurevych, I., and Zesch, T. (2012). Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 435–440. Association for Computational Linguistics.

Baroni, M. and Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Baroni, M., Bernardini, S., Ferraresi, A., and Zanchetta, E. (2009). The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Bartunov, S., Kondrashkin, D., Osokin, A., and Vetrov, D. (2015). Breaking sticks and ambiguities with adaptive skip-gram. *arXiv preprint arXiv:1502.07257*.

Biemann, C. and Riedl, M. (2013). Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95.

Biemann, C. (2010). Co-occurrence cluster features for lexical substitutions in context. In *Proceedings of the 2010 Workshop on Graph-based Methods for Natural Language Processing*, TextGraphs-5, pages 55–59, Uppsala, Sweden.

Bordes, A., Weston, J., Collobert, R., and Bengio, Y. (2011). Learning structured embeddings of knowledge bases. In *Proc. AAAI*, San Francisco, CA, USA.

Brown, S. W. (2008). Choosing sense distinctions for wsd: Psycholinguistic evidence. In *In Proceedings of Association for Computational Linguistics*, pages 249–252, Columbus, Ohio, USA. Association for Computational Linguistics.

Camacho-Collados, J., Pilehvar, M. T., and Navigli, R. (2015). A unified multilingual semantic representation of concepts. *In Proceedings of Association for Computational Linguistics, Beijing, China*.

Erk, K., McCarthy, D., and Gaylord, N. (2009). Investigations on word senses and word usages. In *ACL*, pages 10–18, Suntec, Singapore.

Gurevych, I., Eckle-Kohler, J., Hartmann, S., Matuschek, M., Meyer, C. M., and Wirth, C. (2012). Uby: A large-scale unified lexical-semantic resource based on lmf. In *In Proceedings of EACL*, pages 580–590, Avignon, France.

Hanks, P. (2013). *Lexical analysis: norms and exploitations*. The MIT Press.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *In Proceedings of ACL*, pages 873–882, Jeju Island, Korea.

Jurgens, D. and Klapaftis, I. (2013). Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second joint conference on lexical and computational semantics (* SEM)*, volume 2, pages 290–299.

Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., and Suchomel, V. (2014). The sketch engine: Ten years on. *Lexicography*, 1(1):7–36.

Li, J. and Jurafsky, D. (2015). Do multi-sense embeddings improve natural language understanding? In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP'2015*, pages 1722–1732, Lisbon, Portugal.

Manandhar, S., Klapaftis, I. P., Dligach, D., and Pradhan, S. S. (2010). Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 63–68. Association for Computational Linguistics.

McCarthy, D., Koeling, R., Weeds, J., and Carroll, J.

(2004). Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 279–286, Barcelona, Spain, July.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *Workshop at International Conference on Learning Representations (ICLR).*

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Navigli, R. and Ponzetto, S. P. (2010). Babelnet: Building a very large multilingual semantic network. In *ACL*, pages 216–225, Uppsala, Sweden.

Navigli, R., Litkowski, K. C., and Hargraves, O. (2007). Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 30–35, Prague, Czech Republic. Association for Computational Linguistics.

Neelakantan, A., Shankar, J., Passos, A., and McCallum, A. (2014). Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar, October. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Rothe, S. and Schütze, H. (2015). Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China, July. Association for Computational Linguistics.

Schütze, H. (1998). Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–123.

Tian, F., Dai, H., Bian, J., Gao, B., Zhang, R., Chen, E., and Liu, T.-Y. (2014). A probabilistic model for learning multi-prototype word embeddings. In *COLING*, pages 151–160, Dublin, Ireland.