Правительство Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский университет
«Высшая школа экономики»
Факультет гуманитарных наук

Образовательная программа
«Фундаментальная и компьютерная лингвистика»

КУРСОВАЯ РАБОТА

На тему «Мультиязычные подстановки в задаче определения смысла слова (WSI)»

Тема на английском «Multilingual Substitutes in WSI»

Студентки 3 курса
группы БКЛ-201
Князьковой Виктории Игоревны

Научный руководитель
Сериков Олег Алексеевич
Приглашенный преподаватель
Научный сотрудник Института
Искуственного Интеллекта

Научный консультатнт
Протасов Виталий Павлович
Младший научный сотрудник Института
Искусственного Интеллекта

Москва — 2023

Contents

1. Introduction

Word Sense Induction (WSI) plays a crucial role in natural language processing by addressing the challenge of grouping instances of an ambiguous word based on their intended meanings. Unlike Word Sense Disambiguation (WSD), which focuses on determining the correct sense of a word in a given context, WSI involves categorizing instances of the word into distinct clusters based on their underlying senses. One prominent approach to tackle the WSI task is the lexical substitution approach, which involves generating alternative words that can substitute the target word in a given context. These substitutes are derived using distributional properties and semantic relationships extracted from large corpora and vector space models.

However, most existing research on WSI has primarily concentrated on monolingual data, typically relying heavily on English as the primary language. This singular focus overlooks the wealth of information present in other languages, which could potentially enhance the accuracy and coverage of word sense induction. The idea of incorporating multilingual substitutes stems from the understanding that different languages express semantic relationships and nuances in unique ways. By incorporating substitutes from multiple languages, we can harness the distinctive strengths of each language to improve the quality of word sense induction. This approach allows for the exploration of cross-lingual similarities and differences, leading to a more comprehensive understanding of word senses and their contextual variations.

In this paper, we present a novel approach that integrates multilingual substitutes into the WSI task. Our objective is to investigate the effectiveness of different language combinations, determine the optimal number of substitutes for achieving high performance in WSI, and explore whether the inclusion of substitutes from additional languages can yield improvements over previous results. By conducting a series of experiments and evaluations, we aim to shed light on the potential of multilingual lexical substitution for advancing the field of word sense induction.

2. Related work

Recent studies have made significant contributions to Word Sense Induction (WSI) by employing various techniques. One study utilized neural biLM and symmetric patterns for WSI (Amrami and Goldberg [2018]), while another explored the combination of lexical substitutes in neural WSI (Arefyev et al. [2019]). Additionally, a study (Arefyev et al. [2022]) emphasized the importance of considering the target word and its semantics during lexical substitution. Although not directly focusing on multilingual substitutes, these works provide valuable insights that inform our investigation of Multilingual Language Substitutes in WSI.

3. Proposed method

Our proposed approach for the Word Sense Induction (WSI) task involves utilizing a combination of substitutions from different languages. By leveraging the distinctive strengths and linguistic nuances present in various languages, we aim to enhance the accuracy and coverage of word sense induction. Our approach consists of multiple steps. Firstly, we generate substitutes in different languages, combining them to create a comprehensive substitute set. Next, we vectorize the substitutes and utilize them to cluster word occurrences based on the different meanings of the target word. This multilingual substitution approach enables us to capture a more comprehensive understanding of word senses and their contextual variations.

4. Experimental setup

4.1 *Dataset*

In this study, we utilized the test dataset provided by the SemEval2010 competition, specifically from task 14 (Smith et al. [2017]). We focused solely on the test dataset without utilizing any training data since our model does not involve a training phase. The evaluation metrics were calculated exclusively based on the test sample.
The test dataset comprises 100 target words, evenly distributed between nouns and verbs. Each target word is accompanied by a context consisting of a maximum of three sentences. All instances in the dataset are sourced from the sense-tagged section of OntoNotes (Hovy et al. [2006]). The sentences themselves are news texts, sourced from diverse news sources, such as CNN, ABC and others.

4.2 *Languages*

In our research, we selected five widely studied European languages that possess extensive text resources, enabling effective learning for the language model. These languages include English, French, Spanish, German, and Russian.
These languages belong to 3 different language branches: Italic (French, Spanish), Germanic (German, English) and Balto-Slavic (Russian).
English is considered an analytic language, it tends to have a relatively straightforward semantic structure, with less reliance on inflections. English also has a diverse vocabulary, incorporating loanwords from various languages, which contributes to its semantic richness.
French and Spanish, as Romance languages, are more inflectional in nature. They employ extensive inflections to convey grammatical and semantic information. Morphological changes, such as noun and verb inflections, play a significant role in expressing semantic relationships. These languages often have gender and number agreement, allowing for more precise expression of meaning.
German and Russian, on the other hand, are known for their rich inflectional systems. They feature complex noun declensions and verb conjugations, which contribute to semantic precision and clarity. In these languages, inflections play a crucial role in indicating grammatical relationships and expressing semantic nuances.
Although these languages are quite close to each other and do not show much typological diversity, it is more important for us to have good language models for these languages

4.3 *FastText language models*

In this work we will use embeddings retrieved from FastText language models. FastText language models are a type of text representation that incorporates subword information to capture the meaning of words efficiently. Unlike traditional word embeddings like Word2Vec, FastText represents words as bags of character n-grams, which allows it to handle out-of-vocabulary words and capture morphological information. This approach improves coverage and generalization capabilities, particularly for

languages with rich morphology. FastText models are computationally efficient and provide meaningful representations for both common and rare words. However, FastText models are not designed for generating contextualized word embeddings. That is why we utilize our own methodologies to incorporate contextual information, which will be explained in the subsequent sections.
As soon as we need substitutes on multiple languages, we use fasttext pre-trained word vectors (Grave et al. [2018]). This embeddings were trained using the fastText model on a combination of Common Crawl and Wikipedia data. The training utilized the Continuous Bag-of-Words (CBOW) approach with position-weights, employing a dimensionality of 300. Character n-grams of length 5 were utilized, along with a window size of 5 and 10 negative samples during the training process.

4.4 *Alignment matrices*

As the context is solely available in English and our objective involves generating substitutes in four other languages, it becomes necessary to establish a connection between the context vector extracted from the English model's vector space and the word embeddings from other language models. One potential approach is to utilize aligned word vectors. The concept of word alignment aims to bring similar words from different languages closer to each other in a vector representation, while preserving the inherent relationships between words within the same language.
Fastest provides aligned vectors in the .vec format, but unfortunately, these vectors cannot be directly used to initialize the FastText class in their Python module. Although it is possible to work with them directly or using the gensim module, doing so would mean losing the convenience of a pre-existing, efficient C++ module and, more importantly, losing the ability to process out-of-vocabulary words since .bin files are necessary for that functionality. Hence, to address this limitation, we chose to utilize pre-trained alignment matrices (Smith et al. [2017]). By multiplying each vector with these matrices, we transform them into a shared vector space where different languages can be compared effectively.

5. Substitutes generation

The initial step in the lexical substitution approach for addressing the WSI problem involves generating alternative words suitable for the given context instead of the target word. Typically, this is accomplished by selecting the words that are closest to the target word in the vector space. However, since we utilize a non-contextualized vector model, we need to undertake additional steps before generating context-specific substitutes.
To achieve this, we first obtain the necessary context and acquire its vector representation. Subsequently, we sort the nearest neighbors of the target word based on their proximity to the context vector. This allows us to identify the most appropriate substitutes that align with the given context.
Additionally, our approach requires the translation of the target word into multiple languages. This is necessary due to the limitations of the FastText module, which provides nearest neighbors for words rather than vectors. By translating the target word, we can retrieve the nearest neighbors in the respective language and incorporate them into our substitute generation process.
In the subsequent sections, I will provide a comprehensive description of each of these processes.

5.1 *Context embeddings*

Two strategies were employed to extract the context of the target word: a naive approach and one that considers the part of speech of the surrounding words.

In the naive context extraction, n words were selected from both the left and right sides of the target word, excluding the word itself. However, this approach sometimes resulted in including irrelevant elements such as punctuation marks or articles as part of the context, which do not contribute meaningfully to the context. As a result, an alternative approach was explored.

The selection of context based on part-of-speech tags focuses on including only content words that are more likely to have specific meanings that contribute to the context. In this approach, the entire context is parsed using a tool like Spacy. Similar to the previous method, words are selected within a defined window from the left and right sides. However, in this case, the selected words are filtered based on their part-of-speech tags. Words with tags such as 'PUNCT', 'DET', 'PART', 'X', and 'AUX' are discarded since they are less likely to contribute to the context's meaning.

If, after this filtering procedure, no words remain in the context (which can occur), the context is expanded, and the process is repeated to ensure an adequate amount of meaningful context is captured.

5.2 *Target translation*

To perform word translations, we utilized the Google Translate feature from the deep_translator[1] library. However, we encountered several challenges during this process. Firstly, the translation tool did not consider the part of speech of the words, resulting in some verbs being translated as nouns, and vice versa. Although this aspect was not corrected in our approach, we deemed the proximity of the translated values to be sufficient. This is because the generated substitutes were intended for subsequent clustering rather than for direct word replacements within a context where preserving the correct part of speech would be crucial.

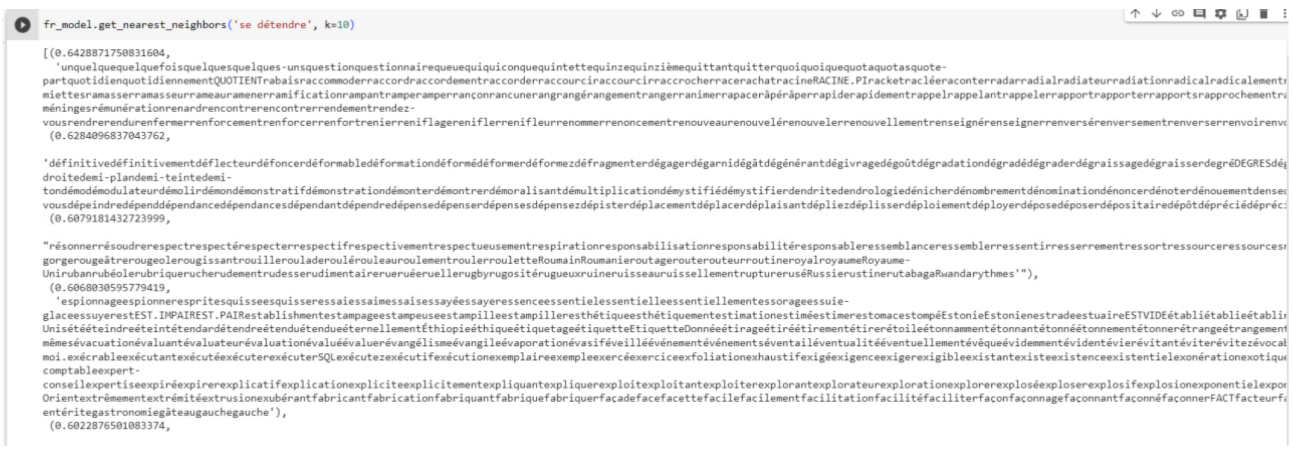
Secondly, we observed that some words were translated as compound phrases. For example, the word "relax" was translated into French as "se détendre", where "se" is a reciprocal particle. Such compound words were treated as out-of-vocabulary words, which in itself was not problematic since FastText can handle them. However, when searching for the nearest neighbors, the initial step involved finding the vector representation of the translated word. This averaged vector was derived from the symbolic n-grams of the compound word, causing it to deviate significantly from the desired value. Consequently, the nearest neighbors retrieved were highly specific words (Fig. 1) that resembled text segments written without spaces.

To avoid generating inaccurate substitutes and preventing the inclusion of such misleading translations, we manually removed these translated phrases and ensured that only one-word translations were retained for these particular words.

5.3 *Non-contextualized substitutes*

Once all the target words are translated (if they are not in English), we proceed to extract the 200 nearest neighbors for each word. Subsequently, a filtering process is applied to refine the obtained list.

[1]https://pypi.org/project/deep-translator/



```
fr_model.get_nearest_neighbors('se détendre', k=10)
```

```
[(0.6428871750831604,
  'unquelquequelquefoisquelquesquelques-unsquestionquestionnairequeuequiquiconquequintettequinzequinzièmequittantquitterquoiquoiquequotaquotasquote-
partquotidienquotidiennementQUOTIENTrabaisraccommoderraccordraccordementraccorderraccourciraccourcirraccrocherracerachatracineRACINE.PIracketracléeraconterradarradialradiateurradiationradicalradicalementr
miettesramasserramasseurrameauramenerramificationrampantramperamperrançonrancunerangrangérangementrangerranimerrapacerâpérâperrapiderapidementrappelrappelantrappelerrapportrapporterrapportsrapprochementra
méningesrémunérationrenardrencontrerencontrerrendementrendez-
vousrendrerendurenfermerrenforcementrenforcerrenfortrenierreniflagereniflerrenifleurrenommerrenoncementrenouveaurenouvelérenouvelerrenouvellementrenseignérenseignerrenversérenversementrenverserrenvoirenvo
 (0.6284096837043762,

'définitivedéfinitivementdéflecteurdéfoncerdéformabledéformationdéformédéformerdéformezdéfragmenterdégagerdégarnidégâtdégénérantdégivragedégoûtdégradationdégradédégraderdégraissagedégraisserdegréDEGRESdég
droitedemi-plandemi-teintedemi-
tondémodémodulateurdémolirdémondémonstratifdémonstrationdémonterdémontrerdémoralisantdémultiplicationdémystifiédémystifierdendritedendrologiedénicherdénombrementdénominationdénoncerdénoterdénouementdensec
vousdépeindredépenddépendancedépendancesdépendantdépendredépensedépenserdépensesdépensezdépisterdéplacementdéplacerdéplaisantdépliezdéplisserdéploiementdéployerdéposedéposerdépositairedépôtdéprécié déprécí
 (0.6079181432723999,

"résonnerrésoudrerespectrespectérespecterrespectifrespectivementrespectueusementrespirationresponsabilisationresponsabilitéresponsableressemblanceressemblerressentirresserrementressortressourceressourcesr
gorgerougeâtrerougeolerougissantrouillerouladeroulérouleauroulementroulerrouletteRoumainRoumanieroutagerouterouteurroutineroyalroyaumeRoyaume-
Unirubanrubéolerubriquerucherudementrudesserudimentairerueruéeruellerugbyrugositérugueuxruineruisseauruissellementrupturerusé RussierustinerutabagaRwandarythmes'"),
 (0.6068030595779419,
  'espionnageespionneresspritesquisseesquisseressaiessaimessaisessayéessayeressenceessentielessentielleessentiellementessorageessuie-
glaceessuyerestEST.IMPAIREST.PAIRestablishmentestampageestampeuseestampilleestampilleresthétiqueesthétiquementestimationestiméestimerestomacestompéEstonieEstonienestradeestuaireESTVIDEétabliétablieétablir
UnisétééteindreéteintétendardétendreétenduétendueéternellementÉthiopieéthiqueétiquetageétiquetteEtiquetteDonnéeétirageétiréétirementétirerétoileétonnammentétonnantétonnéétonnementétonnerétrangeétrangement
mêmesévacuationévaluantévaluateurévaluationévaluéévaluerévangélismeévangileévaporationévasiféveilléévénementévénementséventailéventualitééventuellementévêqueévidemmentévidentévierévitantéviterévitezévocab
moi.exécrableexécutantexécutéexécuterexécuterSQLexécutezexécutifexécutionexemplaireexempleexercéexerciceexfoliationexhaustifexigéexigenceexigerexigibleexistantexisteexistenceexistentielexonérationexotique
comptableexpert-
conseilexpertiseexpiréexpirerexplicatifexplicationexpliciteexplicitementexpliquantexpliquerexploitexploitantexploiterexplorantexplorateurexplorationexplorerexploséexploserexplosifexplosionexponentielexpor
OrientextrêmementextrémitéextrusionexubérantfabricantfabricationfabriquantfabriquefabriquerfaçadefacefacettefacilefacilementfacilitationfacilitéfaciliterfaçonfaçonnagefaçonnantfaçonnéfaçonnerFACTfacteurfa
entéritegastronomiegâteaugauchegauche'),
 (0.6022876501083374,
```

Figure 1: Nearest neighbors to ”se détendre”

As the Fastest model is trained on character n-grams, its vocabulary may include words with typos and punctuation marks attached (Fig. 2). To prevent the inclusion of such words in our final substitutes, we begin by removing any punctuation marks that are incorrectly attached to the words. Next, we perform a pairwise comparison of the combined substitutes and calculate the Levenshtein distance between them. This step aims to eliminate typos and ensure that correctly spelled words are retained in the final substitutes. From each pair, the word with the lower index is selected for removal, taking into consideration that typos are typically less frequent (in FastText, the indexes correspond to word frequency). It is important to note that while this approach does not guarantee a perfect outcome, it does help prevent the substitutes from resembling various misspelled versions of the target word.

By implementing this filtering procedure, we can enhance the quality of the substitutes, ensuring they are more accurate and distinct from the original target word.

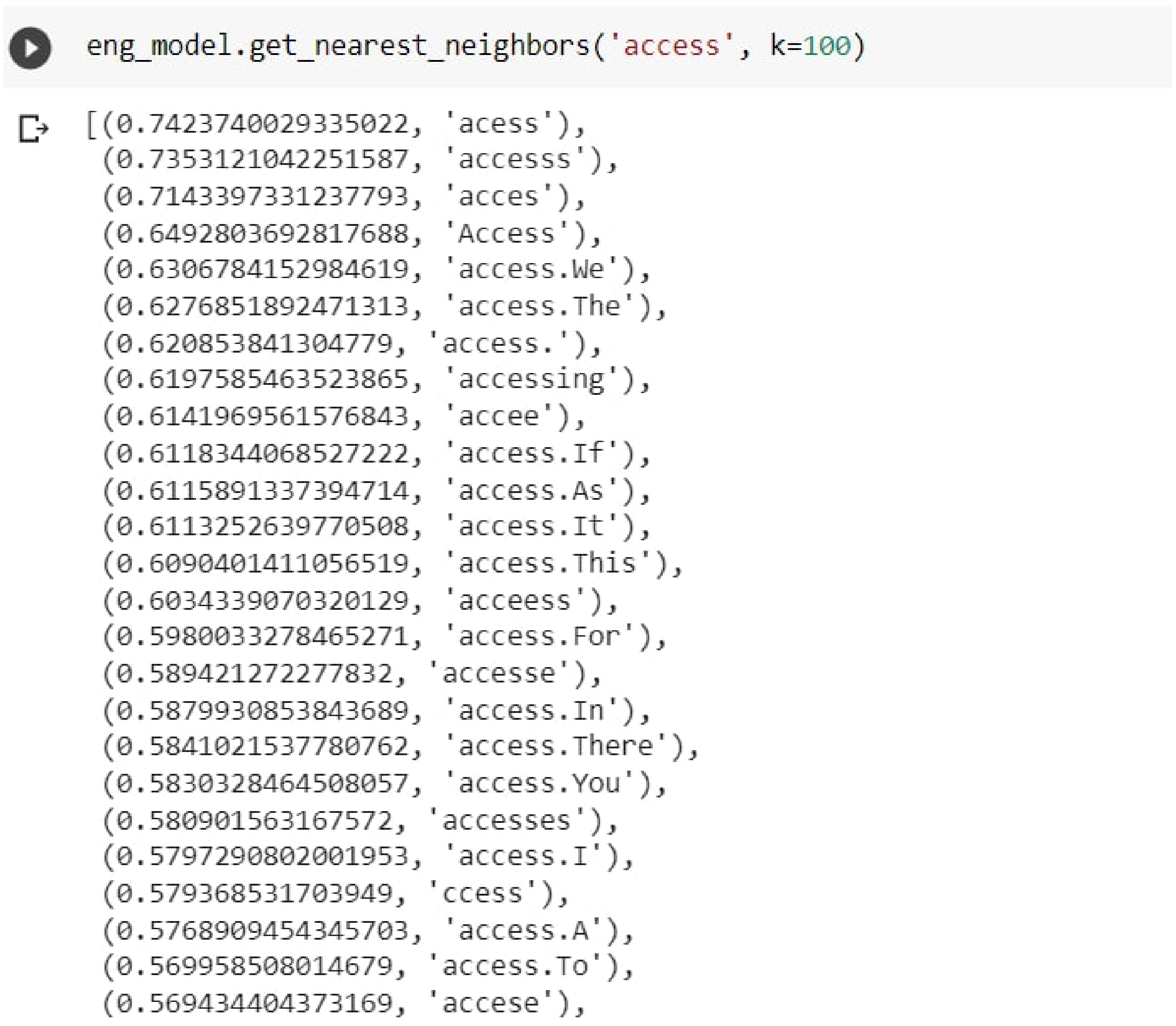
Once the substitutes have undergone the necessary clearance and filtering procedures, the remaining word embeddings are transformed by multiplying them with the mapping matrix. This transformation aligns the embeddings into a shared space, enabling the consideration of their proximity to the context vector.

5.4 *Sorting Substitutes*

Since the substitutions were generated earlier without contextual information, it is now essential to contextualize them appropriately. For each context, the obtained substitutions are sorted based on their proximity to the context vector. This sorting step is crucial for subsequent clustering, as it allows for the selection of the desired top-n substitutions. The sorted results are then stored in separate files for each language. It is worth noting that the generation process for each language was carried out sequentially due to the substantial computational load associated with each model. These separate files will be later combined into a single document during the clustering phase.

6. Clustering

To facilitate the clustering of contexts, an initial step involves constructing documents from the substitutions corresponding to each context. These documents are then transformed into vectors, which are subsequently subjected to clustering using one of the available algorithms.



```
eng_model.get_nearest_neighbors('access', k=100)
```

```
[(0.7423740029335022, 'acess'),
 (0.7353121042251587, 'accesss'),
 (0.7143397331237793, 'acces'),
 (0.6492803692817688, 'Access'),
 (0.6306784152984619, 'access.We'),
 (0.6276851892471313, 'access.The'),
 (0.620853841304779, 'access.'),
 (0.6197585463523865, 'accessing'),
 (0.6141969561576843, 'accee'),
 (0.6118344068527222, 'access.If'),
 (0.6115891337394714, 'access.As'),
 (0.6113252639770508, 'access.It'),
 (0.6090401411056519, 'access.This'),
 (0.6034339070320129, 'acceess'),
 (0.5980033278465271, 'access.For'),
 (0.589421272277832, 'accesse'),
 (0.5879930853843689, 'access.In'),
 (0.5841021537780762, 'access.There'),
 (0.5830328464508057, 'access.You'),
 (0.580901563167572, 'accesses'),
 (0.5797290802001953, 'access.I'),
 (0.579368531703949, 'ccess'),
 (0.5768909454345703, 'access.A'),
 (0.569958508014679, 'access.To'),
 (0.569434404373169, 'accese'),
```

Figure 2: ”access” nearest neighbours

6.1 *Substitutes vectorization*

For vectorizing the substitutions, we employed the tf-idf (term frequency–inverse document frequency) technique. However, a direct combination of the first n substitutions into a single document and applying tf-idf would result in the loss of information regarding their order. Consequently, there would be no distinction between a substitution listed first or twentieth. To address this limitation and account for the proximity of each substitution to the context, we implemented the following approach: within each context, each substitution was replicated a number of times corresponding to its rank after sorting. These duplicates were then stored within a single document. This process was repeated for all contexts, resulting in a collection of documents, the length of which equaled the number of contexts associated with a particular word. Given that this approach is novel and has limited interpretation, we compared its results against those obtained using the conventional tf-idf method, which had been successfully employed in a previous study on the lexical substitution approach in WSI (Arefyev et al. [2022]).

6.2 *Clustering algorithms*

Two distinct clustering algorithms were utilized: agglomerative clustering and the k-means method. The former exhibits faster performance and generates a dendrogram that provides a broader representation of values further broken down into more specific ones. On the other hand, the latter algorithm requires more computation time. However, when the data (i.e., substitution vectors) exhibit a certain level of homogeneity, the k-means method yields more appropriate clusters compared to agglomerative clustering. The latter tends to assign nearly all objects to a single cluster in such cases.

6.3 *Hyperparameter optimization*

Both clustering algorithms require the specification of a hyperparameter: the number of clusters. However, determining the optimal number of clusters for each word is not straightforward. Although the competition dataset provides information about the number of meanings for each word, it would be unfair to rely on external information to achieve the desired outcome. To address this challenge, we have incorporated an automatic selection mechanism that maximizes the silhouette score. When utilizing our clustering class, users have the option to manually define a set of values from which the best one will be chosen, or they can omit the limits, in which case the number of clusters will be selected between 2 and the total number of contexts for each word.

7. Evaluation

For metric evaluation, we utilized scripts provided by the competition organizers. These scripts automatically calculate paired F-score, including precision and recall, as well as v-measure, incorporating measures of homogeneity and completeness. For detailed information on the calculation of these metrics, please refer to the competition article(Manandhar et al. [2010]). Prior to running these evaluation scripts with our results, we had to format the data accordingly. The required format consists of a file where each line contains the target word, context number, and cluster number, separated by spaces (Fig. 8).

8. Experiments and results

We conducted a comprehensive set of experiments involving approximately 60 trials using pre-generated files with substitutes. For each language, two distinct files were utilized: the first using context extraction in a naive manner, and the second incorporating part-of-speech tags in context extraction process.

8.1 *English only LS*

Initially, we focused on clustering English substitutes alone to determine the optimal hyperparameters for subsequent experiments. This involved selecting the context extraction method, substitutes vectorization approach, and clustering algorithm. For vectorization, we employed tf-idf and weighted-tf-idf, which was previously described, as well as the same methods without considering inverse document frequency (idf).

Our experiment revealed that clustering with tf-idf weights, coupled with context collected in a naive manner, yielded the most favorable results. Hence, these parameters were adopted for further

lang	n subst	vecotrizer	clusterizer	context exctraction	fscore	precision	recall	vmeasure	homogenity	completeness	(fs * vm) ** 0.5
en	5	tf	agglomerative	dummy	51.816	71.888	47.930	12.892	21.405	14.484	20.719
en	5	tf	kmeans	dummy	38.580	37.312	50.360	16.968	15.525	22.761	22.248
en	5	tf-idf	agglomerative	dummy	40.870	47.004	49.574	18.343	20.059	25.958	23.180
en	5	tf-idf	kmeans	dummy	24.63	17.189	51.606	20.541	15.911	32.248	21.186
en	5	tf-weighted	agglomerative	dummy	48.490	64.675	49.121	14.836	22.006	18.861	22.241
en	5	tf-weighted	kmeans	dummy	38.775	36.476	52.373	17.936	16.179	23.866	23.702
en	5	tf-idf-weighted	agglomerative	dummy	43.593	51.530	49.506	18.537	21.765	24.116	24.388
en	5	tf-idf-weighted	kmeans	dummy	27.24	20.043	53.145	21.535	17.135	32.605	23.002
en	5	tf	agglomerative	pos_excl	47.959	63.83	48.076	13.822	19.432	17.353	20.904
en	5	tf	kmeans	pos_excl	36.510	34.894	50.528	15.862	14.412	21.785	20.977
en	5	tf-idf	agglomerative	pos_excl	38.068	41.439	50.491	19.311	19.069	27.720	23.724
en	5	tf-idf	kmeans	pos_excl	24.472	18.122	51.935	19.732	15.627	31.186	20.520
en	5	tf-weighted	agglomerative	pos_excl	44.489	54.053	48.656	15.967	19.083	19.936	22.719
en	5	tf-weighted	kmeans	pos_excl	33.428	29.026	50.563	17.671	14.952	25.067	21.548
en	5	tf-idf-weighted	agglomerative	pos_excl	38.187	39.925	50.081	19.634	18.676	28.423	23.857
en	5	tf-idf-weighted	kmeans	pos_excl	25.462	18.445	52.477	20.717	16.184	32.411	21.617

Table 1: WSI average result for english language with 5 substitutes

experiments. In terms of the clustering algorithm, agglomerative clustering exhibited superior performance based on average metrics. However, closer inspection of the clustering results revealed that agglomerative clustering tended to assign almost all contexts to a single cluster, with only a few exceptions (like in Fig. 3). This clustering behavior did not align with our understanding of what an ideal clustering of word contexts should look like, prompting us to predominantly utilize k-means clustering in subsequent experiments. Although the average metric scores were lower for k-means clustering, it demonstrated more stable performance in terms of v-measure and recall.

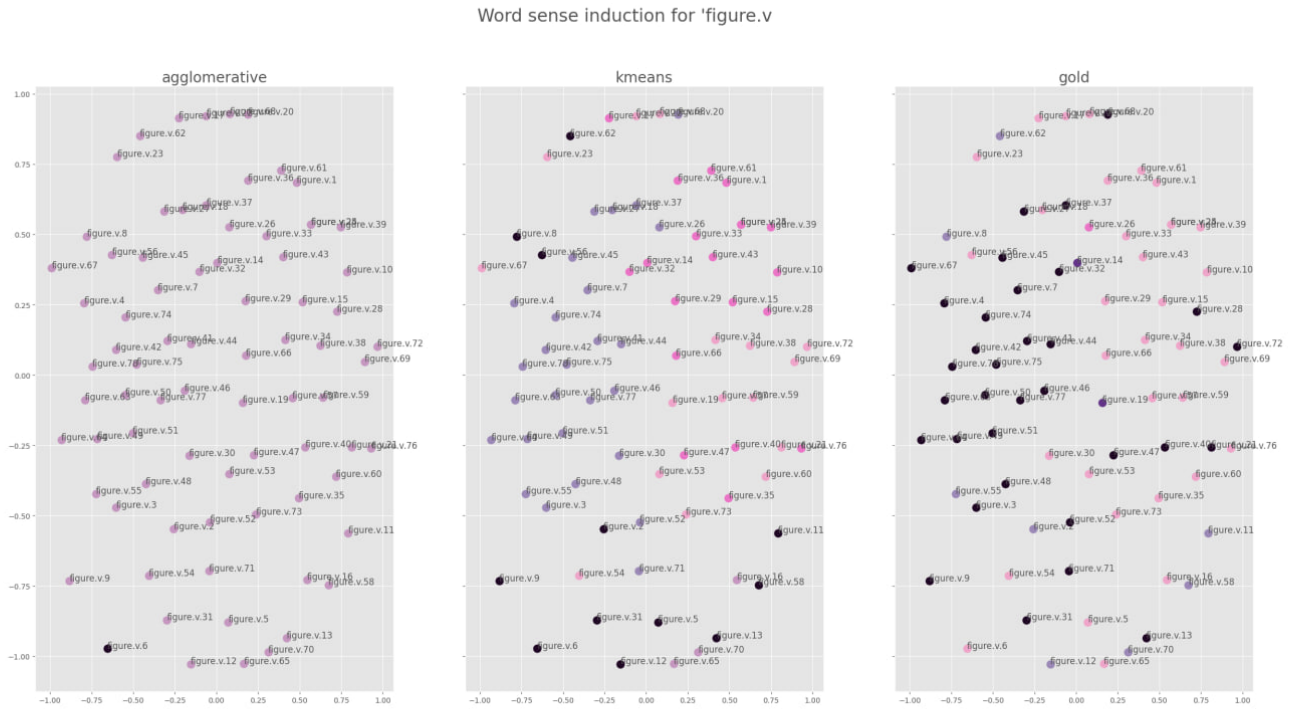


Figure 3: Clusterizations for target word ”figure.v” and comaprison with gold clusterization for that word. The left and the middle clusterization is the result of experiment with following parameters: n_subst=20, languages=fr-es-de-ru-en, vectorizer=tf-idf-weighted, context_extraction=dummy

8.2 *Number of substitutes*

In this experiment we were aimed to establish the optimal number of substitutions required to solve the WSI problem. We examined substitutions in all five languages, considering the first 5, 10, 15, 20, and

25 words (Fig. 4). Results indicated that the best average metric scores were obtained when utilizing 20 substitutions from each language. Furthermore, an analysis revealed that increasing the number of substitutes led to higher f-scores, but lower v-measure values. This trend is likely attributable to the increase in dimensionality of vector representations as the number of substitutions grows, resulting in sparser and more homogeneous matrices, which resemble the clustering behavior observed in agglomerative clustering when most instances are assigned to a single cluster.

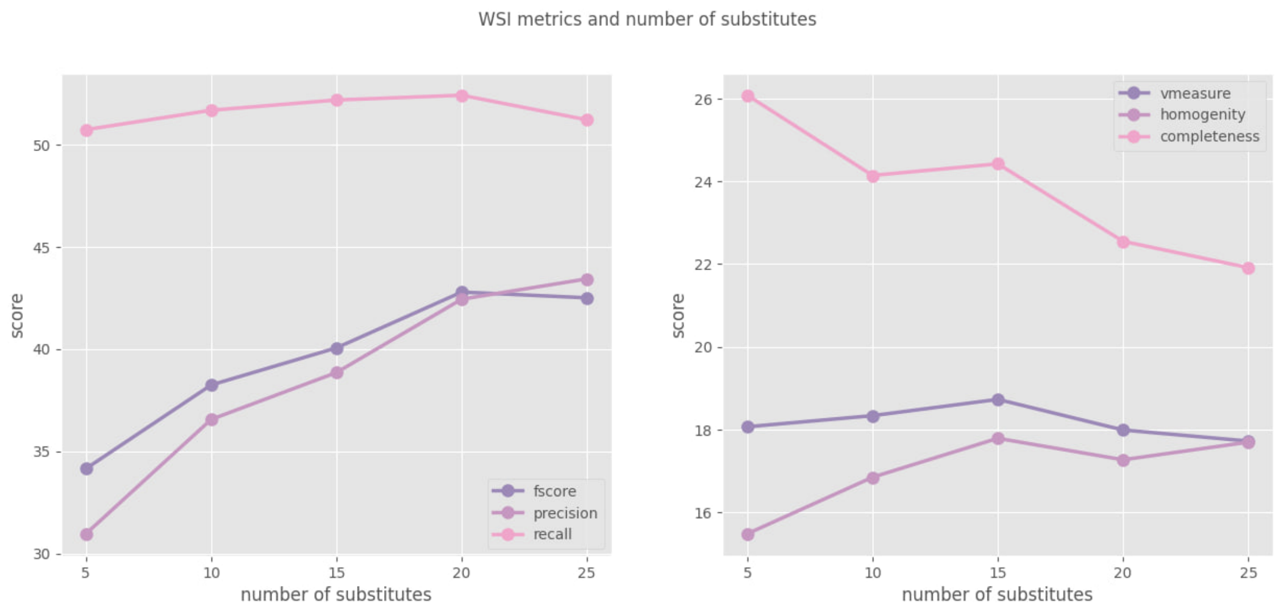


Figure 4: Results of clusterization based on differend number of substitutes for all 5 languages

8.3 *Different languages*

Subsequently, we explored WSI based on monolingual substitutes for each language under consideration. For this experiment, we employed 20 substitutions per language, as determined in the previous experiment. As expected, the most favorable outcomes were obtained when using substitutions in the language of the context, i.e., English and all other languages' substitutes showed much lower result (Fig. 5).

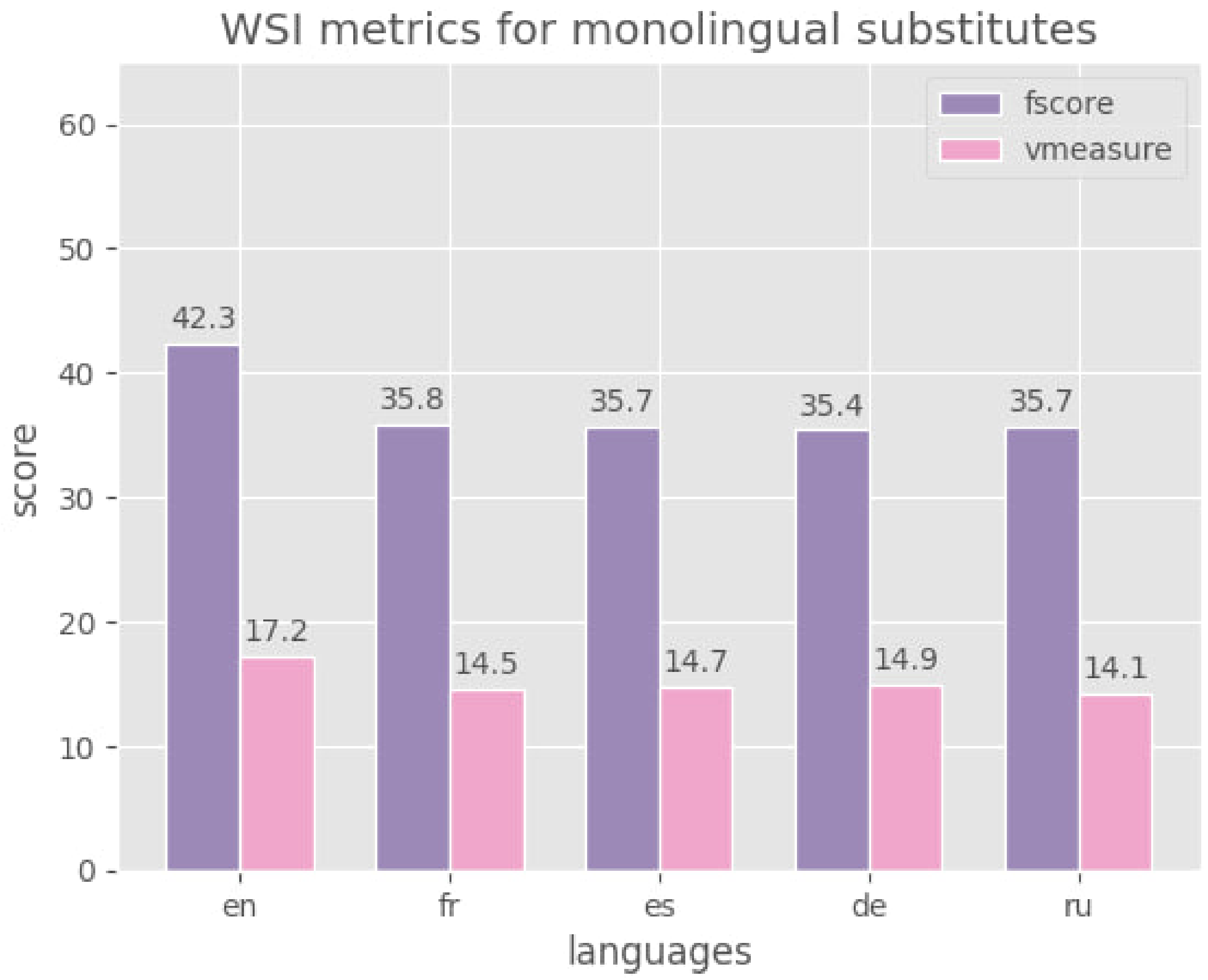


Figure 5: Average WSI results for monolingual substitutes in different languages

8.4 *Different language combinations*

Finally, we systematically explored all possible language combinations to determine the most effective substitutes from different language sets. We selected 20 substitutes from each language and employed the k-means method for clustering. The resulting graph (Fig. 6) shows the top-performing values of our metrics (fscore, vmeasure, and their harmonic mean) grouped by the number of languages used for substitutes, along with the language combinations that yielded these outcomes. Remarkably, nearly all of the best models exhibited some degree of improvement, though marginal, over the sole use of English substitutes for WSI. The best results were obtained by the Spanish-English substitutes and the combined substitutes from all languages, except German.

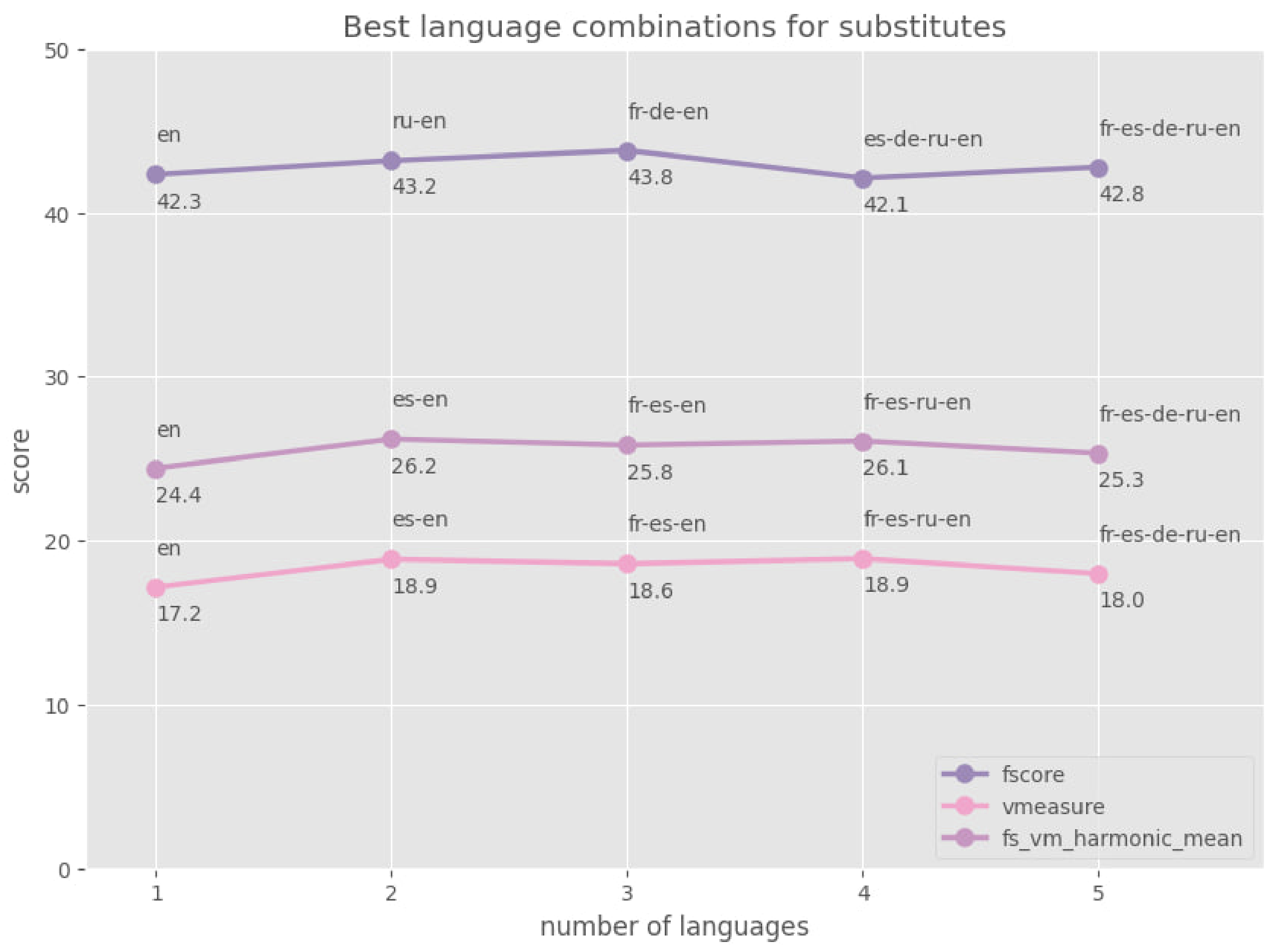


Figure 6: WSI results for the best sets of languages of length n

9. Conclusion

The multilingual lexical substitution approach for the WSI problem shows promise, despite the suboptimal metric scores obtained. Notably, some metrics, such as v-measure, outperformed the results achieved by leading teams in the SemEval-2010 competition. Furthermore, the incorporation of additional languages resulted in a modest improvement in clustering quality. However, it is evident that the use of non-contextualized models like FastText does not suit well for this task. Significant manual intervention and various corrective measures are necessary at each step, including word translation, vector alignment, and contextualization, which ultimately do not significantly enhance the overall quality. Of course, nowadays it is worth using more complex language models based on LSTM or transformers to solve lexical ambiguity. Combining neural networks with multilingual lexical substitutions may hold the potential to achieve a new state-of-the-art result in the WSI task.

References

Asaf Amrami and Yoav Goldberg. Word sense induction with neural bilm and symmetric patterns. *arXiv preprint arXiv:1808.08518*, 2018.

Nikolay Arefyev, Boris Sheludko, and Alexander Panchenko. Combining lexical substitutes in neural word sense induction. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 62–70, 2019.

Nikolay Arefyev, Boris Sheludko, Alexander Podolskiy, and Alexander Panchenko. Always keep your target in mind: Studying semantics and improving performance of neural lexical substitution. *arXiv preprint arXiv:2206.11815*, 2022.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

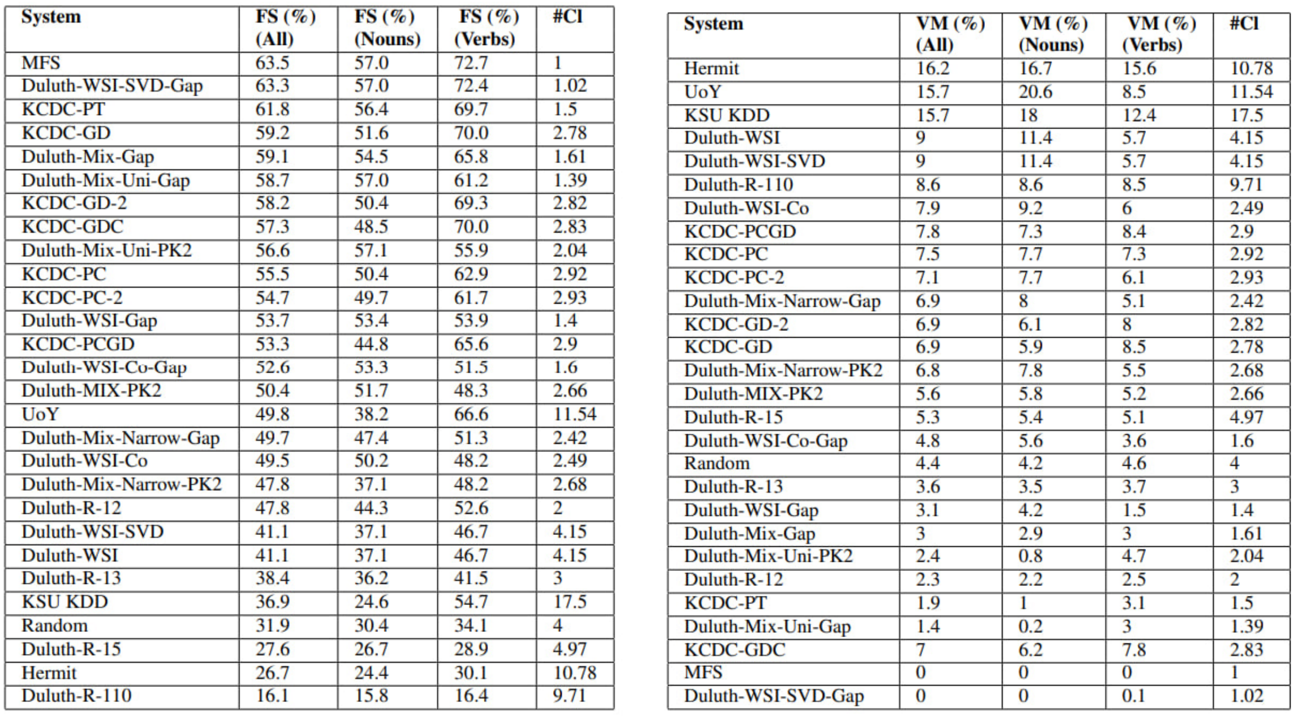
Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, 2006.

Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach, and Sameer Pradhan. Semeval-2010 task 14: Word sense induction &disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 63–68, 2010.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax, 2017.

Appendices

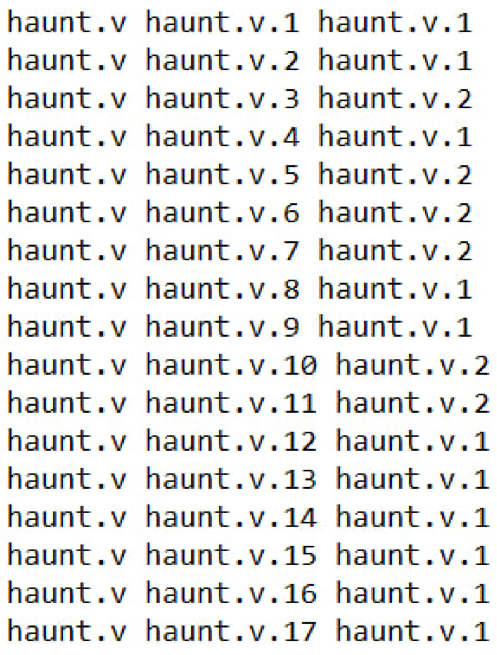
Our github: `https://github.com/vknyazkova/Multilinguial_Substitutes_WSI`



System	FS (%) (All)	FS (%) (Nouns)	FS (%) (Verbs)	#Cl
MFS	63.5	57.0	72.7	1
Duluth-WSI-SVD-Gap	63.3	57.0	72.4	1.02
KCDC-PT	61.8	56.4	69.7	1.5
KCDC-GD	59.2	51.6	70.0	2.78
Duluth-Mix-Gap	59.1	54.5	65.8	1.61
Duluth-Mix-Uni-Gap	58.7	57.0	61.2	1.39
KCDC-GD-2	58.2	50.4	69.3	2.82
KCDC-GDC	57.3	48.5	70.0	2.83
Duluth-Mix-Uni-PK2	56.6	57.1	55.9	2.04
KCDC-PC	55.5	50.4	62.9	2.92
KCDC-PC-2	54.7	49.7	61.7	2.93
Duluth-WSI-Gap	53.7	53.4	53.9	1.4
KCDC-PCGD	53.3	44.8	65.6	2.9
Duluth-WSI-Co-Gap	52.6	53.3	51.5	1.6
Duluth-MIX-PK2	50.4	51.7	48.3	2.66
UoY	49.8	38.2	66.6	11.54
Duluth-Mix-Narrow-Gap	49.7	47.4	51.3	2.42
Duluth-WSI-Co	49.5	50.2	48.2	2.49
Duluth-Mix-Narrow-PK2	47.8	37.1	48.2	2.68
Duluth-R-12	47.8	44.3	52.6	2
Duluth-WSI-SVD	41.1	37.1	46.7	4.15
Duluth-WSI	41.1	37.1	46.7	4.15
Duluth-R-13	38.4	36.2	41.5	3
KSU KDD	36.9	24.6	54.7	17.5
Random	31.9	30.4	34.1	4
Duluth-R-15	27.6	26.7	28.9	4.97
Hermit	26.7	24.4	30.1	10.78
Duluth-R-110	16.1	15.8	16.4	9.71

System	VM (%) (All)	VM (%) (Nouns)	VM (%) (Verbs)	#Cl
Hermit	16.2	16.7	15.6	10.78
UoY	15.7	20.6	8.5	11.54
KSU KDD	15.7	18	12.4	17.5
Duluth-WSI	9	11.4	5.7	4.15
Duluth-WSI-SVD	9	11.4	5.7	4.15
Duluth-R-110	8.6	8.6	8.5	9.71
Duluth-WSI-Co	7.9	9.2	6	2.49
KCDC-PCGD	7.8	7.3	8.4	2.9
KCDC-PC	7.5	7.7	7.3	2.92
KCDC-PC-2	7.1	7.7	6.1	2.93
Duluth-Mix-Narrow-Gap	6.9	8	5.1	2.42
KCDC-GD-2	6.9	6.1	8	2.82
KCDC-GD	6.9	5.9	8.5	2.78
Duluth-Mix-Narrow-PK2	6.8	7.8	5.5	2.68
Duluth-MIX-PK2	5.6	5.8	5.2	2.66
Duluth-R-15	5.3	5.4	5.1	4.97
Duluth-WSI-Co-Gap	4.8	5.6	3.6	1.6
Random	4.4	4.2	4.6	4
Duluth-R-13	3.6	3.5	3.7	3
Duluth-WSI-Gap	3.1	4.2	1.5	1.4
Duluth-Mix-Gap	3	2.9	3	1.61
Duluth-Mix-Uni-PK2	2.4	0.8	4.7	2.04
Duluth-R-12	2.3	2.2	2.5	2
KCDC-PT	1.9	1	3.1	1.5
Duluth-Mix-Uni-Gap	1.4	0.2	3	1.39
KCDC-GDC	7	6.2	7.8	2.83
MFS	0	0	0	1
Duluth-WSI-SVD-Gap	0	0	0.1	1.02

Figure 7: F-score and v-measure results of SemEval-2010 task 14



```
haunt.v haunt.v.1 haunt.v.1
haunt.v haunt.v.2 haunt.v.1
haunt.v haunt.v.3 haunt.v.2
haunt.v haunt.v.4 haunt.v.1
haunt.v haunt.v.5 haunt.v.2
haunt.v haunt.v.6 haunt.v.2
haunt.v haunt.v.7 haunt.v.2
haunt.v haunt.v.8 haunt.v.1
haunt.v haunt.v.9 haunt.v.1
haunt.v haunt.v.10 haunt.v.2
haunt.v haunt.v.11 haunt.v.2
haunt.v haunt.v.12 haunt.v.1
haunt.v haunt.v.13 haunt.v.1
haunt.v haunt.v.14 haunt.v.1
haunt.v haunt.v.15 haunt.v.1
haunt.v haunt.v.16 haunt.v.1
haunt.v haunt.v.17 haunt.v.1
```

Figure 8: Example of output for SemEval task