BOS at LSCDiscovery: Lexical Substitution for Interpretable Lexical Semantic Change Detection

Artem Kudisov[▽]   **Nikolay Arefyev**[◇,▽,△]

[▽]Lomonosov Moscow State University / Moscow, Russia
[△]National Research University Higher School of Economics / Moscow, Russia
[◇]Samsung Research Center Russia / Moscow, Russia
dark.artbeam@gmail.com, nick.arefyev@gmail.com

Abstract

We propose a solution for the LSCDiscovery shared task on Lexical Semantic Change Detection in Spanish. Our approach is based on generating lexical substitutes that describe old and new senses of a given word. This approach achieves the second best result in sense loss and sense gain detection subtasks. By observing those substitutes that are specific for only one time period, one can understand which senses were obtained or lost. This allows providing more detailed information about semantic change to the user and makes our method interpretable.

1 Introduction

LSCDiscovery is a shared task on Lexical Semantic Change Detection (LSCD) in Spanish (D. Zamora-Reina et al., 2022). The participants were provided with two corpora in Spanish, corresponding to 1810-1906 and 1994-2020 respectively, and were asked to solve two subtasks. In the first subtask the participants were asked to rank the given list of about 4K words according to the degree of their semantic change. The second subtask required to determine for each given word if its senses occurring in two corpora are different (and optionally, if it has acquired some new senses, and if it has lost any old ones).

2 Background

Our approach is based on the bag-of-substitutes (BOS) representation of word meaning in context (Başkaya et al., 2013; Arefyev and Zhikov, 2020). Lexical substitutes are those words that can replace a given target word in a given text fragment without making this fragment ungrammatical or substantially changing the meaning of the target word. For ambiguous words, lexical substitutes depend on their meaning expressed in a particular context. For instance, some reasonable substitutes for the word *fly* in the sentence *A noisy fly sat on*

my shoulder are *bug*, *beetle*, *butterfly*, *firefly*, *insect*, etc. But in the sentence *We will fly to London* they are different: *walk*, *run*, *bike*, etc.

In order to generate lexical substitutes, we employ the XLM-R[1] masked language model (Conneau et al., 2020). This model was pre-trained on 2.5T of data in 100 languages as a masked language model, i.e. it received text fragments with some tokens hidden (replaced with the special <mask> token) and was trained to guess those hidden tokens by their context. This kind of pre-training is partially aligned with the lexical substitution task because the model can predict words compatible with the given context. However, there are no guarantees that these words are similar or related by meaning to the target word. Suitable types of lexical substitutes (e.g., synonyms, hypernyms, co-hyponyms) and suitable degree of their similarity to the target word depend on the target task and can be controlled with various techniques explored in (Arefyev et al., 2020). In our solution, we employ the dynamic patterns proposed by Amrami and Goldberg (2018) and explained in 3.2.

Unlike the traditional bag-of-words representation, which contains those words that occur in a text fragment, the BOS representation is built from lexical substitutes. Thus, it better represents the meaning of some specific target word in a given text fragment rather than the whole fragment in general. Clustering of the BOS vectors is a successful approach to solve the Word Sense Induction (WSI) task, i.e. to discover senses of ambiguous words. This approach was explored in many papers, including (Başkaya et al., 2013; Amrami and Goldberg, 2018, 2019; Arefyev et al., 2019, 2020) among others. Also, a substitution-based WSI model was employed to solve the LSCD task in (Arefyev and Zhikov, 2020; Arefyev and Bykov, 2021). However, in our solution we avoid solving the more

---

[1]The pre-trained xlmr.large from fairseq library is used without any fine-tuning.

`<mask>`-(y-T)		`<mask><mask>`-(y-T)		`<mask>`-(incluso-T)	
Substitute	Prob.	Substitute	Prob.	Substitute	Prob.
documentos (documents)	0.367	archivos (records)	0.016	documentos (documents)	0.391
libros (books)	0.160	escritos (letters)	0.012	libros (books)	0.082
datos (data)	0.052	informes (reports)	0.010	datos (data)	0.039
actos (acts)	0.036	dos documentos (two documents)	0.010	textos (texts)	0.037
textos (texts)	0.032	expedientes (records)	0.008	contratos (contracts)	0.014

Table 1: For the word *actas* (*reports*) in *ayer recibimos dos actas literales* (*yesterday we received two verbatim reports*), 5 most probable substitutes with 1 or 2 subwords are shown. The patterns with *y* (*and*) and *incluso* (*including*).

general and probably more difficult WSI task that requires clustering. Instead, we propose methods to directly obtain LSCD predictions from the BOS vectors.

3 Model description

For each target word we sample some examples of its usage from both corpora and generate lexical substitutes for them. Then we build two sets of BOS vectors for old and new examples, describing old and new senses of the word respectively. Finally, the distances from old to new examples are calculated, and their average is returned as the predicted score of graded change. Following previous works on LSCD (Giulianelli et al., 2020; Laicher et al., 2021), we will denote this average as the Average Pairwise Distance (APD). Notice that our vector representation is very different from those works.

For the second subtask, if APD is greater than a certain threshold, we predict that this word has changed its meaning. To determine whether it has acquired new senses and whether any old senses were lost, we propose three different methods based on pairwise distances.

3.1 Collected data

For each target word w_i, we lemmatize[2] both corpora and retrieve all examples with w_i in different grammatical forms. Then we take the same number N_i of examples from the old and the modern set of examples.[3]

3.2 Substitute generation

For each example we generate several types of substitutes with different dynamic patterns, post-

---

[2] We used the Spanish lemmatizer from Spacy proposed by the organizers.

[3] If possible, $N_i = 100$ examples are sampled without replacement from each set. Otherwise, we take all $N_i < 100$ examples from the smaller set and sample the same number of examples from another set.

Pattern	weight
`<mask>`	0.25
`<mask>`-(y-T)	0.25
T-(y-`<mask>`)	0.25
`<mask>`-(incluso-T)	0.0625
T-(incluso-`<mask>`)	0.0625
`<mask>`-(por-ejemplo-T)	0.0625
T-(por-ejemplo-`<mask>`)	0.0625

Table 2: In LS_m1_7, we employ 7 single-subword patterns with *y* (*and*), *incluso* (*including*) and *por ejemplo* (*for example*) with the specified weights.

LS_m1_2 patterns	LS_m2_2 patterns	weight
`<mask>`-(y-T)	`<mask><mask>`-(y-T)	0.5
T-(y-`<mask>`)	`<mask><mask>`-(y-T)	0.5

Table 3: In LS_m1_2 and LS_m2_2 we employ 2 single-subword and 2 two-subword patterns respectively.

process them and combine together to get a single vector representation. Dynamic patterns are similar to the Hearst patterns by nature (Hearst, 1992). They were proposed in (Amrami and Goldberg, 2018) to obtain from masked language models those substitutes that do not only fit the given context, but also are similar or related to the target word by meaning. For instance, using patterns with the Spanish conjunction *y* (English: *and*) we hope to obtain mostly co-hyponyms of the target word, while patterns with the adverb *incluso* (English: *including*) shall bias the model towards generating hypernyms or hyponyms, depending on the position of the target word. Table 1 shows some examples.

Table 2 lists all dynamic patterns we use. All patterns contain the special token `<mask>` that XLM-R is asked to recover, and some of them contain the variable T representing the target word. Given a pattern and an example for some target word, first we replace the target word with this pattern, and then replace the variable T (if any) back with the target word. For simplicity, let us consider an example in English. Given the sentence

We can fly to London and using the pattern `<mask> (and T)`, we first obtain *We can `<mask>` (and T) to London*, and finally have *We can `<mask>` (and fly) to London*.

The vocabulary of XLM-R consists of 250K subwords in 100 different languages, which are sometimes whole frequent words, but most often pieces of words. To better describe word meaning, we generate substitutes consisting of different number of subwords. To achieve this, we apply patterns with several `<mask>` tokens, for instance, `<mask><mask> (y T)`.

To find probable sequences of subwords that could fill the `<mask>` tokens, we apply a slightly modified greedy decoding strategy. For the leftmost `<mask>` token, $topK = 150$ most probable subwords are predicted first. Then for each of those subwords we generate one continuation using greedy decoding. Below we will say that a substitute is not generated for a particular pattern in a particular example if it was not among $topK$ substitutes generated this way. For computational reasons, we generated only substitutes with one or two subwords and did not apply beam search for decoding. Examples of two-subword substitutes are in table 1.

3.3 Substitute post-processing and combination

Next, we post-process all substitutes for each example: convert them to lower case, remove all words except for the last one from multi-word substitutes, apply stemming.[4] After post-processing, we sum the probabilities of duplicated substitutes.

For each example, we combine substitutes generated for different patterns by calculating the weighted average of the corresponding probability distributions. In **LS_m1** and **LS_m2** (Lexical Substitution with one-subword substitutes and two-subword substitutes respectively), for combination we use patterns and weights presented in Tables 2 and 3. The weights were selected based on a few experiments on the development set consisting of 20 words, so these weights are likely suboptimal. It is possible that one of the substitutes is not generated by XLM-R for a certain pattern. In this case, during combination we assume that the corresponding probability is equal to the minimal probability among all substitutes generated for this pattern.

---

[4]The Spanish stemming from nltk.stem.snowball was used.

Model/Team	JSD,SPR	COMPARE,SPR
baselines		
baseline1	**0.543** (4)	**0.561** (3)
baseline2	0.092 (8)	0.088 (6)
best results of other teams		
myrachins	**0.735** (1)	**0.842** (1)
UsrD7	0.702 (2)	0.829 (2)
aishein	0.553 (3)	0.558 (4)
our results		
#LS_m1_7+APD	-0.125 (9)	-0.129 (8)
our post-evaluation results		
LS_m1_7+APD	**0.584** (3*)	0.598 (3*)
LS_m1_2+APD	0.562 (3*)	0.562 (3*)
LS_m2_2+APD	0.576 (3*)	**0.637** (3*)

Table 4: Graded Change Discovery results. # denotes the buggy implementation. * denotes possible ranks of the corresponding results in the leaderboard.

3.4 BOS vectors

For each target word w_i we build $2N_i$ BOS vectors for old and new examples. These vectors are basically bag-of-word vectors built for $topK$ most probable substitutes for each example. Only substitutes that were generated for more than 3% and less than 90% of examples of the target word are taken into account[5].

3.5 Graded Change Discovery

APD (Average Pairwise Distance). After building the BOS vectors, we calculate the cosine distance from each old to each new example, resulting in a matrix of size $N_i \times N_i$. The APD is calculated by averaging all cells in this matrix. Finally, we sort test words according to their APDs and submit their ranks as the predicted change scores.[6]

3.6 Binary Change Detection

For the main Binary Change Detection subtask, if the calculated APD is greater than the certain $threshold$[7], then we predict that this word has changed its meaning. In this case we also try to determine if it has acquired new senses and if it has lost some old ones (sense loss and sense gain detection subtasks). We try three methods to determine that.

---

[5]We used CountVectorizer from scikit-learn, where $min\_df = 0.03$ was selected in range from 0 to 0.05 with 0.01 step and $max\_df = 0.9$ was selected in range from 0.85 to 1 with 0.01 step.

[6]There was a mistake in the original implementation of the ranking procedure. After the competition we fixed it, which significantly improved the results of this method (see table 4 for comparison).

[7]$threshold = 0.8$ was selected on the development set in the range from 0.7 to 0.9 with 0.05 step.

Model/Team	CH, F1	GAIN, F1	LOSS, F1
baselines			
baseline1	0.537 (9)	NaN (8)	NaN (6)
baseline2	0.222 (10)	0.211 (7)	0.000 (6)
best results of other teams			
myrachins	**0.716** (1)	**0.491** (3)	**0.688** (1)
dteodore	0.709 (2)	0.000 (8)	0.000 (6)
rombek	0.687 (3)	0.490 (4)	0.593 (3)
our results			
LS_m1_7+AID	**0.658** (4*)	0.393 (6*)	0.137 (6*)
LS_m2_2+min	0.636 (6*)	0.418 (6*)	**0.610** (2*)
LS_m1_7+perc.	**0.658** (4)	**0.520** (2)	0.600 (2)
our post-evaluation results			
LS_m1_2+AID	0.628 (7*)	0.4 (6*)	0.076 (6*)
LS_m1_2+min	0.628 (7*)	**0.583** (2*)	0.387 (5*)
LS_m1_2+perc.	0.628 (7*)	0.486 (5*)	**0.608** (2*)
LS_m2_2+AID	0.636 (6*)	0.382 (6*)	0.193 (6*)
LS_m2_2+perc.	0.636 (6*)	0.376 (6*)	0.600 (2*)
LS_m1_7+min	**0.658** (4*)	0.533 (2*)	0.564 (5*)

Table 5: Binary Change Detection results. * denotes possible ranks of the corresponding results in the leaderboard.

Model	GAIN,F1	LOSS,F1
LS_m1_2+AID	0.4 (6*)	0.076 (6*)
LS_m1_2+min	**0.583** (2*)	0.387 (5*)
LS_m1_2+perc.	0.486 (5*)	0.608 (2*)
LS_m2_2+AID	0.382 (6*)	0.193 (6*)
LS_m2_2+min	0.418 (6*)	**0.610** (2*)
LS_m2_2+perc.	0.376 (6*)	0.600 (2*)
LS_m1_7+AID	0.393 (6*)	0.137 (6*)
LS_m1_7+min	0.533 (2*)	0.564 (5*)
LS_m1_7+perc.	0.520 (2)	0.600 (2)

Table 6: Comparison of aggregation methods. * denotes possible ranks of the corresponding results in the leaderboard.

4 Experiments and Results

4.1 Phase 1: Graded Change Discovery

In this subtask, it was required to rank about 4K target words according to their degree of semantic change (the higher rank, the stronger change). The final quality of ranking was evaluated for 60 hidden words only by the Spearman's correlation with the gold ranks (Bolboaca and Jäntschi, 2006).

Table 4 provides the results for the first phase. Our original implementation of the ranking procedure had mistakes in the ranking procedure, so the results are poor. After the competition, we fixed the mistake and obtained the correct results, which are comparable to the 3rd best participant in the leaderboard.

LS_m1_2 and **LS_m2_2** differ only in the number of masks in the used patterns. So comparing their scores, we can say that using two-subword substitutes is more preferable than one-subword substitutes. In **LS_m2_7** seven patterns are combined compared to two patters in **LS_m1_2**, this gives a significant improvement despite somewhat arbitrarily selected weights. Developing some principled ways of finding promising dynamic patterns and weights for their combination is a reasonable direction for future work. **LS_m1_7** has a slightly higher JSD,SPR score, but its COMPARE,SPR score is lower and it uses a more complex pattern combination than **LS_m2_2**. A more detailed investigation is presented in Appendix A.

4.2 Phase 2: Binary Change Detection

In this subtask the participants were asked to determine if target words have changed their meanings. And if so, how exactly (have acquired and/or have lost senses). Three F1-scores are calculated: Binary Change Detection (CH), Sense Gain Detection (GAIN), Sense Loss Detection (LOSS). Results are

AID (Average Inner Distance). We calculate APDs between only new examples AID_1 and between only old examples AID_2. If $AID_1 > (AID_2 - b_1)$, we predict that a new sense appeared. If $AID_2 > (AID_1 - b_2)$, we predict that an old sense is lost.[8] Thus, we assume that a difference in average inner distances for two sets of examples indicates that there is a difference in underlying sets of senses.

min. We calculate an $N_i \times N_i$ matrix of pairwise distances from old to new examples and assume that if some new sense appeared, then a new example exists that is far from all old examples. Thus, if there is at least one new example whose minimal distance to the old examples is greater than some $threshold$ [9], we predict that a new sense appeared. Sense loss is determined symmetrically.

perc. (percentile). This is similar to the previous method, but we calculate the 5th percentile instead of the minimum, i.e. we allow at most 5% of examples from the old corpus to be closer to an example of the new sense from the new corpus than the specified threshold. We assume that this should make the model less sensitive to noisy examples and more stable.

---

[8] $b_1 = 0.03$, $b_2 = 0$. These values were selected on the development set in the range from -0.1 to 0.1 with 0.01 step.

[9] $threshold = 0.8$ was selected on the development set in the range from 0.7 to 0.9 with 0.05 step.

presented in Table 5 where we have the 2nd best submission for GAIN and LOSS optional subtasks.

LS_m1_2 + APD and **LS_m2_m2 + APD** have 0.628 and 0.636 CH,F1 scores respectively, which means that using two-subword substitutes is slightly better than one-subword. But in the case of **LS_m1_7 + APD** we already get 0.658 CH,F1 resulting in the 4th rank.

Using **AID** method does result in good GAIN,F1 and LOSS,F1 scores (Table 6). At the same time **min** and **percentile** show a better results but they highly depend on used LS patterns, i.e., in the some cases these methods improves only GAIN,F1 or LOSS,F1 scores, but not both of them.

5 Discriminative substitutes

The main advantage of LS-based models is their interpretability. We can roughly understand word meanings looking at the discriminative substitutes, i.e. the substitutes specific for a particular subset of examples.

Table 7 provides some examples for *disco* (*disc*) and *satélite* (*satellite*). We take old examples O and those new examples M, that were determined by LS_m1+percentile model as being far away from O. Then we find substitutes with the largest ratio $\frac{P(w|M)}{P(w|O)}$[10], i.e. those substitutes that are rarely generated for old examples but frequently generated for new examples that are not similar to any old examples.

Disco (disc)		Satélite (satellite)	
LP	0.72/0.00	CD	1.00/0.00
EP	0.55/0.00	video	1.00/0.00
documentos	0.55/0.00	internet	1.00/0.00
videos	0.50/0.00	televisión	0.88/0.00
mp	0.50/0.00	FM	0.88/0.00
anime	0.44/0.00	señal	0.88/0.00
memoria	0.44/0.00	Internet	0.88/0.00
PC	0.44/0.00	canal	0.88/0.00
USB	0.44/0.00	TV	0.88/0.00
b	0.44/0.00	web	0.88/0.00
MP	0.44/0.00	vídeo	0.77/0.00

Table 7: Discriminative substitutes generated for the `<mask>(y T)` pattern. The probabilities $P(w|M)$ and $P(w|O)$ are shown for each substitute. Documentos is 'documents', señal is 'signal', memoria is 'memory' and canal is 'channel'.

From the Table 7 we can see that *disco* (*disc*) and *satélite* (*satellite*) have acquired new senses as *a data storage device* and *satellite television* respectively.

---

[10]If the word denominator is 0, we demand $P(w|M)$ to be greater 0.2, otherwise we don't consider such word.

6 Efficiency

The set of the target words proposed in Phase 1 was supposed to be a challenge for participants due to its size. For 4385 words given we have collected about 777K examples. Generation of substitutes for all examples took 13 GPU-hours and 310 GPU-hours for each one-mask and two-mask pattern respectively on V100 GPUs. All other steps took incomparably less time.

7 Conclusion

We have proposed an interpretable approach to lexical semantic change detection. This approach shows the 2nd best result for sense loss and sense gain detection subtasks. It provides techniques to understand which senses were obtained or lost by a word.

Acknowledgements

References

Asaf Amrami and Yoav Goldberg. 2018. Word sense induction with neural bilm and symmetric patterns. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4860–4867.

Asaf Amrami and Yoav Goldberg. 2019. Towards better substitution-based word sense induction. arXiv.

Nikolay Arefyev and Bykov. 2021. An interpretable approach to lexical semantic change detection with lexical substitution. In *Computational linguistics and intellectual technologies: Papers from the annual conference Dialogue*, 20.

Nikolay Arefyev, Boris Sheludko, and Tatiana Aleksashina. 2019. Combining Neural Language Models for Word Sense Induction. In *Analysis of Images, Social Networks and Texts*, page 105–121. Springer International Publishing.

Nikolay Arefyev, Boris Sheludko, Alexander Podolskiy, and Alexander Panchenko. 2020. Always keep your target in mind: Studying semantics and improving performance of neural lexical substitution. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1242–1255, Barcelona, Spain.

Nikolay Arefyev and Vasily Zhikov. 2020. BOS at SemEval-2020 task 1: Word sense induction via lexical substitution for lexical semantic change detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 171–179, Barcelona (online). International Committee for Computational Linguistics.

Osman Başkaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. AI-KU: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 300–306, Atlanta, Georgia, USA. Association for Computational Linguistics.

Sorana-Daniela Bolboaca and Lorentz Jäntschi. 2006. Pearson versus spearman, kendall's tau correlation analysis on structure-activity relationships of biologic active compounds. *Leonardo Journal of Sciences*, 5(9):179–200.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Frank D. Zamora-Reina, Felipe Bravo-Marquez, and Dominik Schlechtweg. 2022. Lscdiscovery: A shared task on semantic change discovery and detection in spanish. In *Proceedings of the 3rd International Workshop on Computational Approaches to Historical Language Change*, Dublin, Ireland. Association for Computational Linguistics.

Mario Giulianelli, Marco Del Tredici, and Raquel Fernández. 2020. Analysing lexical semantic change with contextualised word representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3960–3973, Online. Association for Computational Linguistics.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

P. S. Kostenetskiy, R. A. Chulkevich, and V. I. Kozyrev. 2021. HPC resources of the higher school of economics. *Journal of Physics: Conference Series*, 1740:012050.

Severin Laicher, Sinan Kurtyigit, Dominik Schlechtweg, Jonas Kuhn, and Sabine Schulte im Walde. 2021. Explaining and improving BERT performance on lexical semantic change detection. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 192–202, Online. Association for Computational Linguistics.

A Substitute analysis

Our models mostly depend on the used LS patterns and ways of their combination. So it is important to make some investigations about them. In this section we study the following questions.

- Which single-subword pattern gives the best results and how these results depend on the number of substitutes generate (topk)?

- Is it better to use single-subword or multi-subword substitutes?
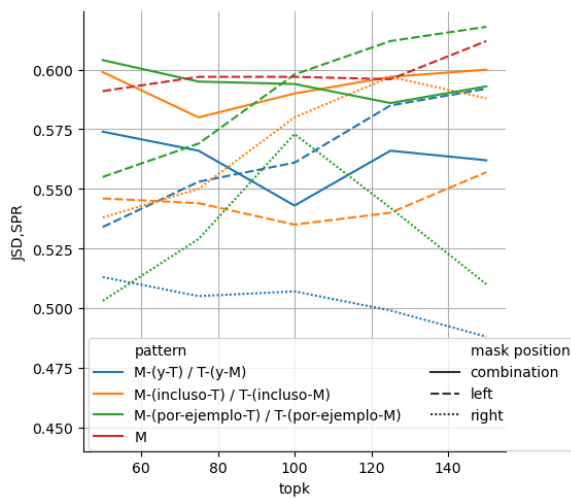
- Do brackets and dashes affect the results?

Figure 1: Dependence of the JSD,SPR score on the pattern and topk.
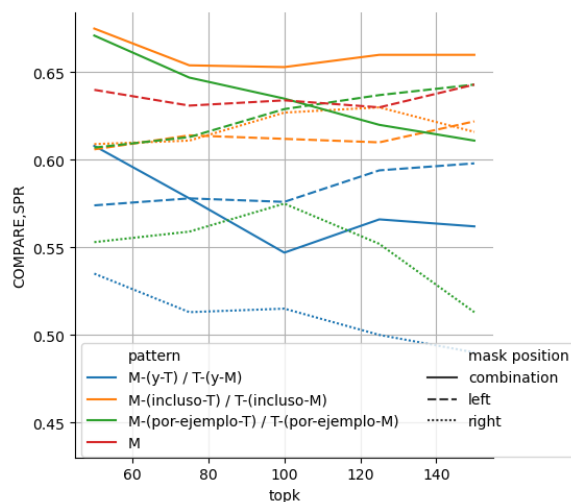
Figure 2: Dependence of the COMPARE,SPR score on the pattern and topk.

For brevity, we will use `M` instead of `<mask>` in the pattern descriptions. In the follow-ing figures `mask position` describes the position of the `<mask>` token. For example, if the pattern is *M (y T) / T (y M)*, `mask position=left` refers to the pattern *M (y T)*, and `mask position=right` refers to *T (y M)*. Finally, `mask position=combination` denotes the combination of these patterns with equal weights.

A.1 One-subword subword

In **LS_m1_7** we use 7 patterns with different weights, which were selected after only a few experiments on the development set. In this section we study how the results depend on the patterns and try to find simpler and more intuitive ways of the substitute combination. Figures 1 and 2 show JSD,SPR and COMPARE,SPR for different patterns.

It is interesting that in all cases the `left` patterns give better results than the `right` ones, except for the incluso-based patterns. Also in all cases the combination averages the results of both patterns, again except for the combination of incluso-based patterns which on the contrary improves the results.
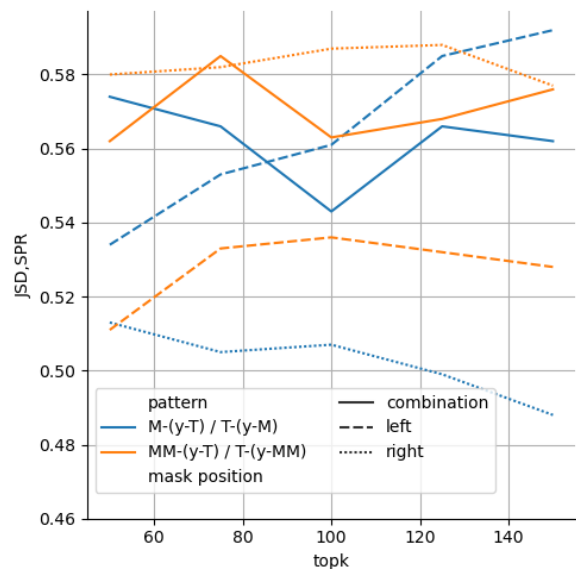

Figure 3: Comparison of one-subword and two-subword substitutes.

A.2 One-subword substitutes vs. two-subword substitutes

We assume that using more masks should improves results because this allows to generate more diverse substitutes. Figure 3 provides comparison of

patterns with different number of masks. As we suspect, using *T (y MM)* pattern gives a much better results than *T (y M)*. However combination of two-mask patterns results in just slightly higher score and one-mask pattern *M (y T)* even outperforms *MM (y T)*.
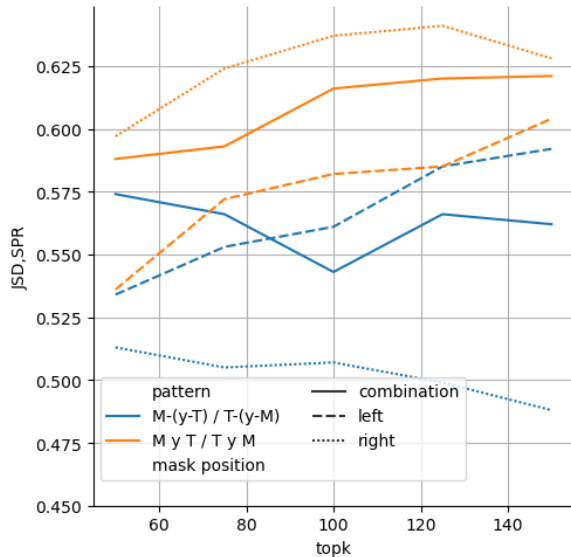

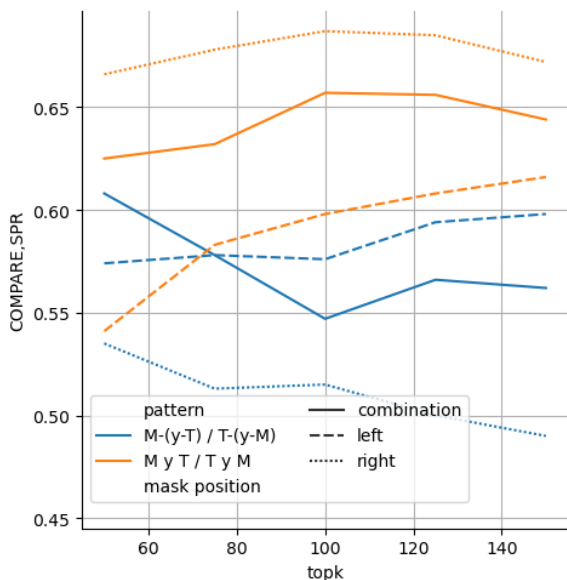Figure 4: Comparison patterns with and without brackets.

Figure 5: Comparison patterns with and without brackets.

A.3 Patterns without brackets and dashes

In the patterns discussed above we have extra dashes which were added by mistake and potentially could affect the results, so firstly we remove

them from patterns. Also we have assumption that using brackets is not common thing in Spanish so such patterns could spoil generated substitutes and final results. To prove it we decide to compare y-based patterns with and without brackets and dashes.

In the Figures 4 and 5 we can see that in all cases refusal to use brackets and dashes improves our results quite well, especially the right pattern get around 0.1 growth in JSD,SPR and COMPARE,SPR scores.