Automatic Generation of *Cloze* Question Stems

Rui Correia[1,3], Jorge Baptista[2], Maxine Eskenazi[3], and Nuno Mamede[1]

[1] INESC-ID Lisboa / IST, Portugal
[2] Universidade do Algarve, Portugal
[3] Language Technologies Institute, Carnegie Mellon University, USA
{Rui.Correia,Nuno.Mamede}@inesc-id.pt, jbaptis@ualg.pt, max@cs.cmu.edu

Abstract. *Fill-in-the-blank* questions are one of the main assessment devices in REAP.PT tutoring system. The problem of automatically generating the stems, i.e. the sentences that serve as basis to this type of question, has been studied mostly for English, and it remains a challenge for a language as morphologically rich as European Portuguese (EP), for which additional data scarcity problems arise. To address this problem, a supervised classification technique is used to model a classifier that decides whether a given sentence is suitable to be used as a stem in a *cloze* question. The major focus is put in the feature engineering task, describing both the development of new criteria, and the adaptation to EP of features already explored in the literature. The resulting classifier filters out inadequate stems, allowing experts to build and personalize their instruction focusing on a set of potentially good sentences.

Keywords: Question Generation, *Cloze* Questions, CALL.

1 Introduction

REAP.PT [9] (REAder-specific Practice for PorTuguese) is the Portuguese version of REAP [7], developed at Carnegie Mellon University. This Computer Assisted Language Learning (CALL) tutoring system aims at teaching vocabulary to L2 learners of European Portuguese (EP) having reading activities as a starting point. Students are presented with real texts, collected from the Web, in which a set of words from the Portuguese Academic Word List [2] (P-AWL) are highlighted. After each reading, there is an assessment phase, composed of questions targeting the vocabulary that was highlighted in the text.

Also known as *fill-in-the-blank*, *cloze* questions are one of the question types used in that assessment phase, testing the highlighted words in context by requiring the student to find the word that better fits a sentence. *Cloze* questions are composed of three elements: **stem** (sentence from which a word was deleted), **target word** (correct answer), and **distractors** (set of wrong answers).

To successfully create an adequate *cloze* question, the stem and the distractors must be in accordance, so that no ambiguity allows for more than one acceptable answer. Correia et al. [3] focused on generating coherent distractors for a set of 4,000 stems manually selected by linguists. This set of stems was produced according to a predefined set of criteria such as considering only full sentences and

H. Caseli et al. (Eds.): PROPOR 2012, LNAI 7243, pp. 168–178, 2012.

not fragmentary text, excluding paratextual elements or lexicographic context, excluding sentences where the target word is at the beginning or end, considering only sentences with length between 100 and 200 characters, and choosing sentences that constitute a non ambiguous environment for the correct identification of the target word. For each word in P-AWL, two or three stems were selected. Figure 1, taken from Pino et al. [13], exemplifies an inadequate stem, showing that not considering some of these criteria can lead to an undesired question formulation, where all the options fit in the blank slot.

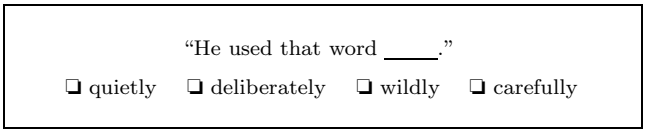

Fig. 1. Example of an inadequate *cloze* question formulation

Relying on experts to generate this resource is expensive, time-consuming, and dependent on the level of expertise and common sense of each individual. Additionally, each time the target word list suffers changes or if REAP.PT is adapted to a more specific context (e.g. teaching medicine vocabulary), the process of generating stems manually has to be repeated. These issues motivated the development of automatic techniques to generate stems for *cloze* questions in the context of REAP.PT.

The remainder of this document is structured as follows: Section 2 presents some previous work, Section 3 describes the proposed solution, Section 4 presents the evaluation of the solution and the results achieved, and finally, Section 5 concludes and proposes future work. For better comprehension, all the Portuguese examples were translated to English, maintaining the aspects that they are intended to represent.

2 Related Work

One of the first attempts to automatic generation of *cloze* question stems came from Hoshino and Nakagawa [8], who trained a model to decide where to insert blank spaces, given a text. In their work, they used machine learning tools (Naive Bayes and K-Nearest Neighbors), a gold standard of questions for training, and a set of features (such as part-of-speech of the previous/next words, position of the target word in the sentence, and sentence length). The authors were able to successfully insert a blank 60% of the time, concluding that more features needed to be considered in order to achieve better results.

In the context of vocabulary tutoring systems, Pino et al. [13] generated stems using two different methods: a baseline technique, that directly extracts the example sentences from the WordNet [5] and a technique that uses linguistic features to decide on the suitability of a sentence from raw text corpora. These features include the length of the sentence, the number of clauses, co-occurrence

scores and the grammaticality score given by the parser. The authors then manually tuned the weights of each feature on training data. The second method outperformed the baseline, producing high quality stems 66% of the time (an improvement of 26%). The authors stressed the importance of measuring the number of clauses, claiming that stems should be comprised of at least two clauses: one with the target word, and another to specify the context.

Finally, Skory and Eskenazi [16] used crowd-sourcing to prove the significance of the previously mentioned co-occurrence feature, finding a high correlation between that criterion and the number of correct answers given by native speakers.

3 Architecture of the Solution

The solution here proposed merges the main ideas mentioned in the previous section, using machine learning techniques to automatically classify a set of sentences extracted from real text corpora, and considering a set of features that mimic the experts' decisions.

Figure 2 presents the general architecture of the stem generation system. A corpus of real texts of European Portuguese is split into individual sentences and indexed. These sentences are then manually classified as positives or negative examples, forming a gold standard that will be used to train the model. A feature engineering task takes place in parallel, extracting information from the sentences that can act as predictors of sentence quality. The gold standard and the features are then used in a Support Vector Machine (SVM), producing the final stem classifier. The remainder of this section will focus on each one of these modules, giving particular emphasis to the feature computation task.

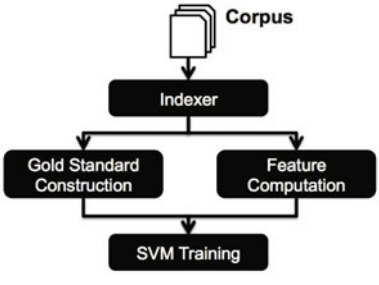

Fig. 2. General architecture of the solution

3.1 Corpus

CETEMPúblico[1] was used as a source of candidate stems. This corpus provides sentences with high quality since they have been extracted from newspaper articles. CETEMPúblico [15] is composed of about 7 million sentences collected from a Portuguese daily newspaper, between 1991 and 1998.

[1] `http://www.linguateca.pt/cetempublico/` (visited in Jan. 2012).

3.2 Indexer

In order to index the sentences from the corpus, the Apache Lucene™[6] text search engine was used. The resulting index allows searching for P-AWL's target words in CETEMPúblico's sentences, for which stems have to be generated, and it also provides an efficient way to compute some of the features (see Section 3.4). In Information Retrieval notation, each sentence of the corpus is a *document*, and each word is a *term*.

3.3 Gold Standard Construction

The motivation behind building a gold standard, and do not use the set of stems manually selected by linguists, has to do with the fact that this set only contains positive examples, i.e, it does not contain the type of sentences that are supposed to be filtered out by the final classifier. In order to build a model that is able to accomplish the task at hand, it is necessary to have both positive and negative instances, not biasing the classifier's decisions (for instance, by using only sentences already selected based on the length criterion).

Thus, all the sentences with the word *computer* (chosen because it had a substantial representation on the corpus, with over 2K examples) were annotated as being good or bad stems, by someone with deep knowledge of the task. It is important to notice that having as a training set sentences targeting only one word is not limiting, since the goal is to build a word-independent classifier.

As one would expect, the number of negative examples (2,057) is higher than the number of positive (180). For the held-out set, we reserved 30 negative and 30 positive examples, randomly chosen. The remaining 150 positive examples and a set of 150 randomly chosen negatives formed the train/test set.

3.4 Feature Computation

The success of a classification task highly depends on the features that are used and the information each one of them is able to represent. In this scope, feature engineering demands for Natural Language Processing (NLP) techniques capable of representing information that the experts use while selecting stems.

Before focusing in each feature, it is important to introduce the NLP tool that supports the computation of most of those criteria. The STRING NLP chain (Figure 3) endows each sentence with information resultant from each of its 4 modules:

- **LexMan** [10, 4] – assigns to each word all the possible POS tags;
- **RuDriCo** [12] – disambiguates the results from LexMan by applying transformation rules based on pattern matching;
- **MARv** [14] – statistic disambiguation of the results from RuDriCo;
- **XIP (Xerox Incremental Parser)** [1] – appends information of elementary syntatic constituents (*chunks*), syntactic dependencies, named entities and anaphoric relations.

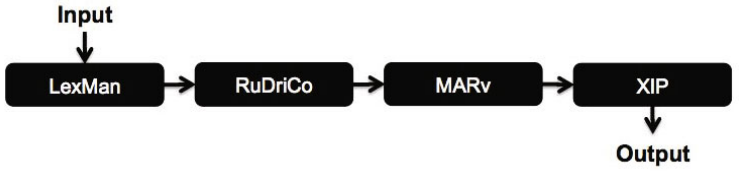

Fig. 3. Simplified STRING chain schema

The remainder of this section describes each one of the features that were used.

Sentence Length – Pino et al. [13] stresses that short sentences do not usually provide sufficient context to be used as stems in a *cloze* question, while long sentences may distract the student by adding unnecessary complexity.

Target Word Position – Additionally, Pino et. al [13] states that the position of the target word, and therefore the blank space, in the beginning or end of the example sentence, is typically an indicator of an inadequate stem (see Figure 1).

Secondary Idea and Secondary Idea Length – These two innovative features take into consideration that sometimes the target word is used in a complementary idea to the main sentence, ultimately representing only side information as in *"Through a services control center (**computer**), you can combine several call centers"*.

However, it may happen that the text between the secondary idea markers (in this case, the brackets) is able to define a context by itself as in *"He is dedicated to computational physics (physics research through **computer** simulation)"*. So, considering the length of the secondary idea is also important.

In Enumeration – Another criterion first explored in this work is the presence of the target word in a list, as an enumeration of objects, actions, etc. Typically, this particular sentence structure does not provide the necessary context. The *COORD* dependency provided by the STRING chain signals this structure.

Proper Names, Foreign Words, Acronyms and Numerical Expressions – Another innovation of the present work is the use of these phenomena. This set of criteria aims at penalizing sentences that require specific domain knowledge to be understood, such as in *"The Hollywood requires a personal **computer** IBM PS/2 or compatible with 80286 processor, operating with the DOS 3.3 operating system or higher and with Microsoft Windows 3.0"*. When these phenomena appear several times in one sentence the student may become distracted with the interpretation of these elements, instead of focusing in solving the exercise. It is worth noting that this feature was not part of the original set of criteria used by the linguists to build the aforementioned set of 4K sentences. It resulted instead from the need to discard the significant amount of sentences in CETEMPúblico with overwhelming content similar to the example above.

Co-occurrences – The literature that focus on stem's generation uses context windows around the target word as the main feature. The approach we present here formulates this feature in a different manner, focusing on the sentence as a whole, and not limiting the co-occurrences window.

The graph-based ranking model TextRank [11] has been used in Keyword Extraction to identify in a text the terms that best represent it. This concept could be used in our scope, finding sentences in which the extracted keywords were the words that should be tested. However, the computational cost of the algorithm and the insufficient context that a single sentence provides, excludes this as a solution.

To mimic TextRank's effect, the Skip Bigram Co-occurrence Counts (SBCC) was computed between each content word of the sentence and the target word. As the name states, this formulation considers occurrences of two words, that are not necessarily contiguous, and is computed according to the following:

$$SBCC(s,k,V) = \sum_{i=0}^{length(s)} f(s_i,k,V) \tag{1}$$

$$f(s_i,k,V) = \begin{cases} \frac{counts(s_i \cap k,V)}{counts(w,V)} & \text{if } s_i \text{ is a content word;} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The SBCC score for a given sentence, s, a target word k, and a set of sentences V, is the sum of the quotient between the number of times a content word, s_i, and the target word k occur in the same sentence in the corpus, and the number of times s_i occurs by itself.

If s_i always occurs in the same sentences that k, the value of f achieves its maximum, i.e., 1. Additionally, since s_i occurs at least once with k, f is never 0. To account for misspells, we assumed that if a word only occurs once in the corpus, it will not be considered in the computation.

In practice, to compute this score, we used the POS tags resulting from the STRING chain to determine the content words, and the index described in Section 3.2 to find the required counts.

Verb Domain – The *VDOMAIN* dependency (representing verb phrases in STRING chain) provided a measure of the sentence's complexity.

Level, Known Words and Unknown Words – In order to take into account the global difficulty of the stem in comparison with the target word that it aims to test, a bag-of-words approach was used to compute grade levels. Based on a corpus of 47 text books, exercise books and national exams, divided by grade levels, the algorithm considers unigrams to estimate the probability of a word being in a certain level. The probability of a word, w_i, being in the level l_j is:

$$P(w_i = l_j) = \frac{counts(w_i, l_j)}{\sum_{k=1}^{N_j} counts(w_k, l_j)} \tag{3}$$

with N_j being the number of different words in the level j.

This method computes a probability for each level (in our case, from 5 to 12) and, looking at the first level in which the word appears and the following 2 levels, assigns the level in which the probability is higher, amongst these three levels. However, this solution may not be entirely adequate in some cases. For instance, if a word occurs only once in the 5^{th} level and then only occurs in the 10^{th}, 11^{th} and 12^{th} levels it will probably be incorrectly classified in the 5^{th} level, even if it appears more often in the higher levels.

This feature filters out candidate sentences where the level of the words in the stem is higher than the level of the target word itself, which is intuitively a bad formulation for a *cloze* question. On the other hand, this criterion boosts sentences with "easier" words to test target words of higher level of difficulty.

Contrary to previous solutions, the number of known and unknown words was also used as a feature. An unknown word was considered to be a word that was not found in any of the textbooks, exercise books or national exams.

3.5 SVM Training

The WEKA data mining tool[2] was used with the *LibSVM* classifier (Support Vector Machines), with a 20-fold cross-validation, using a radial kernel and a cost value of 1,000, increasing the cost of misclassifying points (parameters adjusted using the held-out set). WEKA provides the result of the classification of an instance along with the correspondent probability estimate, allowing for the ranking of the results.

4 Results

The results of the present work are divided in two subsections. Section 4.1 will present the main results of the classifier, while Section 4.2 presents an evaluation of the stems selected by linguists using the resulting classifier.

4.1 General Results

The first experiment used two features that already have been proved in literature to be relevant for the task: *length* and *co-occurrences*. This constitutes a baseline that will allow to compare the contribution of the remaining features.

Figure 4 presents the distribution of the *length* and the *co-occurrence* features, along the manual classification as good or bad stems.

Regarding the *length* criterion, one can see a weak distinction. However, it is interesting to notice that negative examples tend to spread more along the *length* axis, whereas good examples tend to have a lower standard deviation from the mean length. For the *co-occurrences* criterion, it is clear that higher values of this feature tend to constitute a good stem. The majority of the negative examples have a value between 0 and $\frac{1}{4}$ in the Y axis.

[2] `http://www.cs.waikato.ac.nz/ml/weka/` (visited in Jan. 2012).

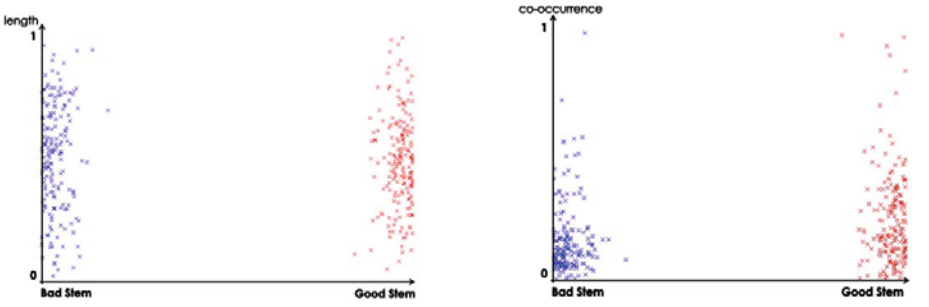

Fig. 4. Distribution (with jitter) of the features *length* **(a)** and *co-occurrence* **(b)**

Table 1 presents the results for the classification task with only two features.

Table 1. Results of the classifier using the *length* and *co-occurrences* criteria

	TP	FP	Precision	Recall	F-Measure
Bad Stem	66.7	31.1	68.2	66.7	67.4
Good Stem	68.9	**33.3**	67.4	68.9	68.1
AVG	67.8	32.2	67.8	67.8	**67.8**

With a *f-measure* of 67.8%, the classifier behaves better than simple chance. However, the percentage of false positives is higher than it would be desirable. 60 sentences were classified as good stems when they were bad stems. Nevertheless, this is still a good result since one is interested in the best few good sentences, instead of complete precision.

Table 2 presents the results when all the features were used, and Table 3 shows the confusion matrices comparison between the first and second approaches. With an increase of 7.7% of the *f-measure*, the inclusion of the new features improved the classifier. The rate of false positives decreased to 21.7% (11.6% lower than in the first attempt).

While analyzing the *information gain* of the features, the *co-occurrence* criterion proved to be the most relevant with a score of 0.2379, followed by *length* with 0.0364. These two features were the ones that were considered in the first experiment, and were pointed out in previous work as good estimates. The only criterion that the classifier dismissed was the *level* feature.

Another interesting result was that 23% of the good stems of the gold standard had a probability estimate greater than 90%. The sentence that ranked higher was *"They stole my laptop* ***computer****, full of my company's files, personal texts and personal e-mail"*. The majority of the false positives were caused by enumerations undetected by the STRING chain and by the fact that the level criterion was discarded, since these two aspects were considered while building the gold standard.

Table 2. Results of the classifier using all criteria

	TP	FP	Precision	Recall	F-Measure
Bad Stem	78.3	27.2	74.2	78.3	76.2
Good Stem	72.8	**21.7**	77.1	72.8	74.9
AVG	75.6	24.4	75.6	75.6	**75.5**

Table 3. Confusion matrix of the classifier using only the *length* and *co-occurrence* features **(a)** and using all criteria **(b)**

		Prediction Outcome	
		Bad Stem	Good Stem
Actual Value	Bad Stem	120	**60**
	Good Stem	56	124

a)

		Prediction Outcome	
		Bad Stem	Good Stem
Actual Value	Bad Stem	141	**39**
	Good Stem	49	131

b)

In order to conclude on the utility of the classifier, it was tested for the noun *office* and for the verb *release*. For the first ten higher ranked sentences, 7 were good stems. The top result for the word *office* was *"Instead of opening a private* ***office****, Maria worked in the Military Hospital but was dismissed for political reasons"* and for the word *release* was *"One of his first acts, already in power, was to* ***release*** *three dozen political prisoners, including former President Olusegun Obasanjo"*.

4.2 Classifying Linguists' Stems

When the stems developed by linguists were submitted to the classifier only 13% of them were classified positively. When looking at the data, the reason why several of those sentences were rejected became clear, and two conclusions could be drawn.

The lowest ranked stems (the ones with high probability of being negative examples) did not respect the set of criteria that was used by the classifier. Some sentences had the target word in the end of the sentence, some had many numbers and acronyms and some were too short or too long. Some examples of these phenomena are *"Allotting,* ***proportionally****, the potential voters of the AD between PSD and PP, the votes in both parties is 32.3 and 4.1% respectively"* or *"Santana Lopes* ***assists*** *TVI"*. This confirmed the motivation of this work: experts' time should be applied on this task on sentences that are already filtered and flagged as potential good stems, reducing errors generated by the exhaustive process of selecting sentences from scratch.

However, the rejected 87% of the stems are far from being composed only of mistakes. The way we applied the classifier turned out to be inappropriate. The classifier should be used to classify stems one target word at a time, instead of

classifying stems from different words in one passage. The manually generated set of stems is composed of sentences that aim to test a great variety of words (the ones in the P-AWL set), containing two or three examples for each inflected form of the target word. These inflected forms will have lower co-occurrence scores than words that are more common, as *computer*, and, when normalized, will be hindered by higher occurring words.

5 Conclusions and Future Work

This work presented a method to generate stems for *cloze* questions, using a supervised machine learning technique. The features developed here proved to be word-independent, and able to select good stems for words not represented in the train set.

Despite being able to reduce the number of false positives as the number of features increased, some criteria could have been used as a filter instead of being a parameter to the SVM (such as the presence of an enumeration). Additionally, the classifier could benefit if the computation of the *co-occurrence* feature took into consideration the lemma of the word (instead of considering the inflected forms themselves).

This work also identified some problems with the set of sentences developed by experts, contributing for the motivation of having automatic techniques for stem generation.

Being only interested in 3 to 5 sentences per word, this resource can help teachers build *cloze* question exercises in an expedite way, using their expertise to focus on a few sentences that are potentially good to be used as stems. In this setup, there is also the advantage of bootstrapping the model as teachers identify good suggestions while REAP.PT is being used.

Acknowledgments. This work was partially supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds and by FCT project CMU-PT/HuMach/0053/2008. The authors would also like to thank Professors Bruno Martins and Pável Calado for the discussion throughout the execution of this work.

References

[1] Ait-Mokhtar, S., Chanod, J., Roux, C.: A Multi-Input Dependency Parser. In: Proceedings of the Seventh International Workshop on Parsing Technologies, pp. 17–19 (2001)

[2] Baptista, J., Costa, N., Guerra, J., Zampieri, M., de Lurdes Cabral, M., Mamede, N.: P-AWL: Academic Word List for Portuguese. In: Computational Processing of the Portuguese Language, pp. 120–123 (2010)

[3] Correia, R., Baptista, J., Mamede, N., Trancoso, I., Eskenazi, M.: Automatic Generation of Cloze Question Distractors. In: Proceedings of the Workshop on Second Language Studies: Acquisition, Learning, Education and Technology, Tokyo, Japan (September 2010)

[4] Diniz, C.: Um Conversor Baseado em Regras de Transformação Declarativas. Master's thesis, Instituto Superior Técnico–Universidade Técnica de Lisboa, Lisbon, Portugal (2011)
[5] Fellbaum, C.: WordNet: An electronic lexical database. The MIT press (1998)
[6] Hatcher, E., Gospodnetic, O.: Lucene in action (2004)
[7] Heilman, M., Collins-Thompson, K., Callan, J., Eskenazi, M.: Classroom success of an Intelligent Tutoring System for lexical practice and reading comprehension. In: Ninth International Conference on Spoken Language Processing, Citeseer (2006)
[8] Hoshino, A., Nakagawa, H.: A Real-Time Multiple-Choice Question Generation for Language Testing: A Preliminary Study. In: Proceedings of the Second Workshop on Building Educational Applications Using NLP, pp. 17–20. Association for Computational Linguistics (2005)
[9] Marujo, L., Mamede, N., Lopes, J., Trancoso, I., Pino, J., Eskenazi, M., Baptista, J., Viana, C.: Porting REAP to European Portuguese. In: International Workshop on Speech and Language Technology in Education, ISCA, Citeseer, Wroxall Abbey Estate, Warwickshire, UK (September 2009)
[10] Medeiros, J.: Processamento Morfológico e Correcção Ortográfica do Português. Master's thesis, Instituto Superior Técnico–Universidade Técnica de Lisboa, Lisbon, Portugal (1995)
[11] Mihalcea, R., Tarau, P.: TextRank: Bringing Order Into Texts. In: Proceedings of the Empirical Methods on Natural Language Processing Conference, pp. 404–411. Association for Computational Linguistics, Barcelona (2004)
[12] Paulo, J.: PAsMo-Pós Analisador Morfológico. Ph.D. thesis, Instituto Superior Técnico–Universidade Técnica de Lisboa, Lisbon, Portugal (2001)
[13] Pino, J., Heilman, M., Eskenazi, M.: A Selection Strategy to Improve Cloze Question Quality. In: Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains. 9th International Conference on Intelligent Tutoring Systems, Citeseer, Montreal, Canada, pp. 22–32 (2008)
[14] Ribeiro, R., Oliveira, L., Trancoso, I.: Using Morphossyntactic Information in TTS Systems: Comparing Strategies for European Portuguese. In: Computational Processing of the Portuguese Language, p.195 (2003)
[15] Santos, D., Rocha, P.: Evaluating CETEMPúblico, a Free Resource for Portuguese. In: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, pp. 450–457. Association for Computational Linguistics (2001)
[16] Skory, A., Eskenazi, M.: Predicting Cloze Task Quality for Vocabulary Training. In: Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications, pp. 49–56. Association for Computational Linguistics (2010)