Implementation of a Slogan Generator

Polona Tomašič*
XLAB d.o.o.
Pot za Brdom 100, 1000 Ljubljana
Slovenia
polona.tomasic@xlab.si

Martin Žnidaršič*
Jožef Stefan Institute
Jamova cesta 39, 1000 Ljubljana
Slovenia
martin.znidarsic@ijs.si

Gregor Papa*
Jožef Stefan Institute
Jamova cesta 39, 1000 Ljubljana
Slovenia
gregor.papa@ijs.si

Abstract

Generation of slogans for companies, products or similar entities is a creative task that is difficult to automate. In this paper we describe our attempt of tackling this problem by combining computational linguistics, semantic resources and genetic algorithms.

Introduction

Use of computers for support or automation of tasks in creative industries is on the rise. Several such tools and methods emerged in recent years for various problems. Generation of slogans is one of the less supported problems in this field. There are some online tools available[1], which seem to use templating and provide results of such a kind. To the best of our knowledge, there is only one scientific study dedicated particularly to slogan (and other creative sentences) generation, namely the BRAINSUP framework (Özbal, Pighin, and Strapparava 2013). The BRAINSUP approach emphasises user's control of the generation process. Namely, by user-provided keywords, domain, emotions and similar properties of the slogans, the user has a lot of control over the generation process. This is practically very useful, as it shrinks the huge search space of slogans and improves the quality of results. In our work, on the other hand, we aim at a completely autonomous approach, which is not influenced by the user in any way, apart from being provided by a short textual description of the target entity. In this paper, we present our current approach, which follows the BRAINSUP framework, but also deviates from it with several modifications. At the core of our slogan generation procedure we use a genetic algorithm (GA) (Bäck 1996), which ensures good coverage of the search space, and a collection of heuristic slogan evaluation functions.

*Authors are affiliated also to the Jožef Stefan International Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia. This research was partly funded by the European Union, European Social Fund, in the framework of the Operational Programme for Human Resources Development, by the Slovene Research Agency and supported through EC funding for the project ConCreTe (grant number 611733) and project WHIM (grant number 611560) that acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission.

[1]http://slogan4u.com ; http://www.sloganizer.net/en/

Resources

Our slogan generation method requires some specific resources, such as a collection of frequent grammatical relations. Here we list these resources, describe their acquisition methodology and provide some illustrative examples.

Database of existing slogans

The database of exisitng slogans serves as a basis for the initial population generation and for comparison with generated slogans. There is a large number of known slogans for different companies and products available online and there are specialized Web pages that contain collections of slogans. However, none of those sources contain all the necessary information, so we constructed our own database in which each instance consists of: slogan, company/product name, official Web site URL and Wikipedia site URL. Currently the database contains 1041 slogans. Here is an example instance: ["Just do it.", "Nike", "http://www.nike.com/", "http://en.wikipedia.org/wiki/Nike"].

Database of frequent grammatical relations

Frequent grammatical relations between words in sentences were used in some of our processes. For their acquisition we used the Stanford Dependencies Parser (Marneffe, MacCartney, and Manning 2006). Stanford dependencies are triplets containing two words, called *governor* and *dependent*, and the name of the relation between them. The parser also provides part-of-speech (POS) tags and phrase structure trees.

To get representatives of frequent grammatical relations between words, we parsed 52,829 random Wikipedia pages, sentence by sentence, and obtained 4,861,717 different dependencies. Each dependency consists of: name of the relation, governor, governor's POS tag, dependent, dependent's POS tag and the number of occurrences.

Database of slogan skeletons

All the gathered known slogans were parsed with the Stanford Dependencies Parser. Grammatical structure of each slogan, without the content words, was then stored in a database. Each skeleton contains information about each position in the sentence - its POS tag and all its dependency relations with other words in the sentence. For example, skeleton of the slogan "Just do it" is [[['advmod', '***',

'VB', '***', 'RB'], ['2', '1']], [['dobj', '***', 'VB', '***', 'PRP'], ['2', '3']]]. Here the first part tells us that the first word (RB - adverb) is *adverbial modifier* of the second word (VB - verb), and the second part indicates that the third word (PRP - pronoun) is a *direct object* of the second word.

Slogan generation

In this section we describe our slogan generation approach in terms of its inputs, outputs and algorithmic steps.

INPUT consists of two items: (1) a textual description of a company or a product, and (2) the algorithm parameters: evaluation function weights, mutation and crossover probabilities, size of the initial population and maximal number of genetic algorithm iterations.

OUTPUT is a set of generated slogans.

ALGORITHMIC STEPS are the following:

1. Parse the input text for keywords and the main entity.
2. Generate the initial population from random skeletons.
3. Evaluate the slogans and select parents for reproduction.
4. Produce a new generation using crossover and mutations.
5. Repeat steps 3. and 4. until predetermined quality of slogans or maximal number of iterations is achieved.

Extraction of keywords and the main entity

This first step is achieved using the Nodebox English Linguistics library[2]. The main entity is obtained by selecting the most frequent entity in the whole text using *nltk* library (Bird, Klein, and Loper 2009).

Example of the keywords and the entity, extracted from the Coca Cola Wikipedia page:
keywords = ['win', 'produce', 'celebrate', 'using', 'marketing', 'north', 'likely', 'drink', 'century', 'diet', 'production', 'root', 'product', 'beverage', 'water', 'image', 'sugar',... ']
entity = 'Coke'

Generation of the initial population of slogans

The procedure of generating the initial population of slogans is based on the BRAINSUP framework (Özbal, Pighin, and Strapparava 2013), with some modifications and additions. It follows these steps:

1. Select a random slogan skeleton from the database.
2. Choose an empty position, which has the largest number of dependency relations in the sentence. Find the set of all possible fillers for that position. Fillers are words from the database of all grammatical relations between words and must satisfy all predefined dependencies and POS tags.
3. Find the intersection between the set of all possible fillers and the set of keywords. If the obtained set is not empty, choose a random word from it and fill the empty position. In case of an empty intersection, choose random word from the 20% of most frequent possible fillers, and fill the empty position.
4. Repeat steps 2 and 3 until all the empty spots are filled.
5. Check if the generated slogan contains any entities. If it does, replace them with the company entity.
6. Repeat steps from 1 to 5 until the initial population of the predetermined size is built.

[2]http://nodebox.net/code/index.php/Linguistics

Evaluation of slogans

To order the slogans by their quality, an aggregated evaluation function was constructed. It is composed of 10 different sub functions, each assessing a particular feature of a slogan with scores in the interval [0,1]. Parameter of the aggregation function is a list of 10 weights that sum to 1. They define the proportions of sub functions in the overall score.

2-gram function In order to work with 2-grams, we obtained the data set of 1,000,000 most frequent 2-grams and 5000 most frequent words in Corpus of Contemporary American English [3](COCA). The 2-gram evaluation score should to some degree represent the relatedness between words in slogan. We assume that slogans containing many frequent 2-grams, are more likely to make sense. The 2-gram evaluation score is computed in the following manner:

1. Assign a score to every 2-gram in the slogan:
   - if 2-gram is among most frequent 2-grams: score = 1,
   - else if one word is an entity and the other is among 5000 most frequent words: score = 0.75,
   - else if one word is among 5000 most frequent words and the other is not: score = 0.5,
   - else score 0
2. Sum the scores of all 2-grams and divide it by the number of all 2-grams in the slogan.

length function This function assigns score 1 to slogans with less than 8 words, and score 0 to longer ones.

diversity function The diversity function evaluates a slogan by counting the number of repeated words. The highest score goes to a slogan with no repeated words. If a slogan contains identical consecutive words, it receives score 0.

entity function It returns 1, if slogan contains the main entity, and 0, if it doesn't.

keywords function If one up to half words in a slogan belong to the set of keywords, the keywords function returns 1. If a slogan doesn't contain any keyword, the score is 0. If more than half of the words in the slogan are keywords, the score is 0.75.

word frequency function This function prefers slogans with many frequent words, as we assume that slogans which contain a lot of infrequent words are not good. The score is obtained by dividing the number of frequent words by the number of all words in the slogan. Word is considered to be frequent, if it is among 5000 most frequent words in COCA.

polarity and subjectivity functions To calculate the polarity and subjectivity scores based on the adjectives in the slogan, we used the *sentiment* function from *pattern* package for Python (De Smedt and Daelemans 2012). We also integrated the *weight* score from SentiWordNet (Baccianella, Esuli, and Sebastiani 2010), which assigns to each word three sentiment scores: positivity, negativity, objectivity.

[3]Davies, Mark. (2011) N-grams data from the Corpus of Contemporary American English (COCA). Downloaded from http://www.ngrams.info on April 15, 2014.

semantic relatedness function This function computes the relatedness between all pairs of content words in the slogan. Stop words are not taken into account. Each pair of words gets a score based on the path distance between corresponding synsets in WordNet (Miller 1995). The final score is the sum of all pairs' scores divided by the number of all pairs.

structure function During the crossover and mutation phase slogans get deformed and can violate grammatical relations requirements. To avoid unusual grammatical structures in slogans, we parse each new slogan with the Stanford Parser and count the number of infrequent POS tags of word phrases in the parse tree. E.g., the POS tag SBAR (subordinating conjunction), represents only around 3% of all word phrases in English texts. If the number of these POS tags is high, the structure score is low.

Production of a new generation of slogans

A list of all generated slogans is ordered descending with regard to the evaluation score. The best 10% of them are all chosen for reproduction. The other 90% of parent slogans are selected uniformly at random.

A new generation is built by pairing parents and performing the crossover function followed by the mutation function which occur with probabilities $p_{crossover}$ and $p_{mutation}$ respectively. Offspring are then evaluated and compared to the parents, in order to remove very similar ones. Remaining slogans proceed to the next generation. These steps are repeated until a generation of slogans reaches the predefined quality score, or the predefined maximal number of iterations is achieved.

Crossover There are two types of crossover functions, the *big* and the *small* one. Both inspect POS tags of the words in both parents, and build a set of possible crossover locations. Each element in the set is a pair of numbers. The first one provides a position of crossover in the first parent and the second one in the second parent. The corresponding words must have the same POS tag. Let the chosen random pair from the set be (p, r). Using the *big* crossover, the part of the first parent, from the p^{th} position forward, is switched with the part of the second parent, from the r^{th} position forward. For *small* crossover only the p^{th} word in the first parent and the r^{th} word in the second parent are switched. Examples for *big* and *small* crossover are in Figure 1.

Mutation Two types of mutations are possible. Possible *big* mutations are: deletion of a random word; addition of an adjective in front of a noun word; addition of an adverb in front of a verb word; replacement of a random word with new random word with the same POS tag.

Small mutations are replacements of a word with its synonym, antonym, meronym, holonym, hypernym or hyponym. Functions for obtaining such replacements are embedded into the Nodebox English Linguistics library and are based on the WordNet lexical database (Miller 1995).

Deletion of similar slogans Every generated slogan is compared to all its siblings and to all the evaluated slogans from the previous generation. If a new child is equal to any other slogan, it gets removed. If more than half of child's words are in another slogan, the two slogans are considered similar. Their evaluation scores are being compared and the one with the higher rate remains while the other one is removed. The child is also removed, if it contains only one word or if it is longer than 10 words. Deletion of similar slogans is our addition to the basic genetic algorithm. It prevents the generated slogans to converge to the initial ones.

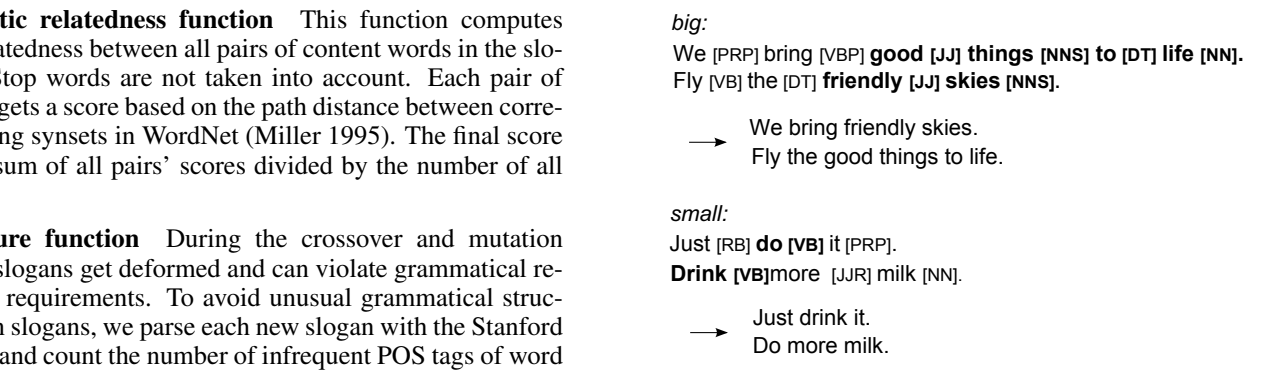

Figure 1: Examples for a *big* and a *small* crossover.

Experiments

We made a preliminary assessment of the generator with experiments as described in the following.

Experimental setting

In presented experiments and results we use a case of Italian luxury car manufacturer Ferrari. The input text was obtained from Wikipedia[4].

First, we tried to find the optimal weights for the evaluation function. We tested different combinations of weights on a set of manually evaluated slogans. The comparison of the computed and the manually assigned scores showed that the highest matching was achieved with the following weights: [2-gram: 0.2, length: 0.04, diversity: 0.05, entity: 0.08, keywords: 0.2, frequent words: 0.07, polarity: 0.08, subjectivity: 0.08, semantic relatedness: 0.05, structure: 0.15].

The probabilities for crossover and mutation functions had to be high so that new generations would not be to similar to previous ones. Probabilities used in our experiments were $p\_big\_crossover = 0.6$, $p\_small\_crossover = 0.9$, $p\_big\_mutation = 0.8$, $p\_small\_mutation = 0.6$. These control parameters were set according to the results of testing on a given input text, as their combination empirically leads to convergence.

Due to the high computational complexity of our method, the maximal number of iterations and the maximal size of initial population were 50 and 20. We performed 20 runs for the same input parameters.

[4]http://en.wikipedia.org/wiki/Ferrari on April 29, 2014.

Table 1: Statistics of slogan scores for 10 best final slogans for all 20 runs. (F = Final, IP = Initial Population)

	min	max	average	median	st. deviation
1	0.785	0.888	0.848	0.85	0.032
2	0.817	0.905	0.849	0.847	0.022
3	0.786	0.896	0.832	0.826	0.034
4	0.780	0.895	0.825	0.809	0.040
5	0.777	0.884	0.837	0.837	0.036
6	0.795	0.937	0.830	0.818	0.039
7	0.773	0.884	0.822	0.812	0.037
8	0.795	0.908	0.833	0.815	0.038
9	0.809	0.894	0.842	0.837	0.029
10	0.789	0.917	0.821	0.816	0.035
11	0.796	0.902	0.844	0.840	0.031
12	0.738	0.902	0.817	0.802	0.051
13	0.761	0.904	0.810	0.772	0.045
14	0.759	0.834	0.789	0.782	0.025
15	0.761	0.901	0.816	0.802	0.042
16	0.816	0.900	0.859	0.861	0.028
17	0.779	0.891	0.831	0.829	0.031
18	0.785	0.888	0.844	0.854	0.035
19	0.739	0.883	0.801	0.787	0.054
20	0.792	0.892	0.834	0.819	0.035
avg. F	0.782	0.895	0.829	0.821	0.036
avg. IP	0.6	0.75	0.66	0.65	0.048

Results and discussion

All 20 runs of the algorithm on the same input data had similar statistical results. Statistics of slogan scores of 10 best final slogans for each run are gathered in Table 1. The score average of slogans increased with each iteration. Table 2 shows its progress.

Table 2: The average increase of the average slogan scores after 10, 20, 30, 40 and 50 iterations.

10	20	30	40	50
21.5%	31.5%	34.7%	37.1%	39.3%

The numbers in both tables show that our method ensures higher slogan scores with each new iteration of genetic algorithm, for a given experimental case. Examples of slogans for one specific run of the algorithm are listed in the following two lists. The first one contains 10 best rated initial slogans and the second one contains 10 best rated final slogans. Evaluation scores are in the brackets.

Initial population:

1. Ferrari is body without substance (0.706)
2. The development of Ferrari (0.696)
3. She swam to make They pay (0.695)
4. increasing production to their output (0.686)
5. allow you with stockings (0.678)
6. causing a Ferrari Saturday (0.676)
7. He wins a role and takes on role (0.66)
8. A successful business to wish (0.631)
9. A success for every artist (0.622)
10. Ferrari uses In his Ferrari (0.599)

Final slogans:

1. make The great meaning of Ferrari (0.905)
2. Ferrari is valuable role with every successful closer (0.865)
3. make you these red Ferrari (0.852)
4. Ferrari is in your largest entertainment more (0.85)
5. only allow you we and Ferrari Saturday (0.848)
6. only make it without its Ferrari (0.847)
7. get The largest being more (0.842)
8. Ferrari is worthy substance closer (0.838)
9. a bright Ferrari Saturday (0.832)
10. They takes The turning more (0.817)

The analysis of initial populations and final slogans in all runs shows that the majority of slogans have grammatical mistakes. This is due to the *big* crossover and the *big* mutation functions. Our system currently lacks an evaluation function for detection or correction of these mistakes.

Some seemingly good slogans can be found already in the initial populations. The evaluation function seems not yet aligned well with human evaluation, as such slogans often do not make it to the final round.

Conclusion

The proposed slogan generation method works and could be potentially useful for brainstorming. The genetic algorithm ensures that new generations of slogan candidates have higher evaluation scores. The critical part of the method is the evaluation function, which is inherently hard to formalize and needs further improvement. We believe that the refinement of semantic and sentiment evaluation functions would increase the quality of slogans, not only their scores.

There are also many other ideas for the future work that would improve the quality of slogans. One is checking for grammatical errors and correcting them if possible. In mutation phase there is a possibility of replacing one word with a whole new word phrase. New weights could be also computed periodically with semi-supervised learning on manually assessed slogans.

References

Baccianella, S.; Esuli, A.; and Sebastiani, F. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proc. of LREC 2010*.

Bäck, T. 1996. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python*. O'Reilly Media.

De Smedt, T., and Daelemans, W. 2012. Pattern for Python. *Journal of Machine Learning Research* 13:2063–2067.

Marneffe, M. D.; MacCartney, B.; and Manning, C. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC 2006*.

Miller, G. A. 1995. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM* 38:39–41.

Özbal, G.; Pighin, D.; and Strapparava, C. 2013. BRAINSUP: Brainstorming Support for Creative Sentence Generation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 1446–1455.