ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ, ОСНОВАННОЕ НА ВЫЧИСЛЕНИИ ЛЕКСИЧЕСКОГО СХОДСТВА, ДЛЯ ВЫРАВНИВАНИЯ ПАРАЛЛЕЛЬНЫХ ТЕКСТОВ И ЕГО ОЦЕНКА[1]

DYNAMIC PROGRAMMING BASED ON LEXICAL SIMILARITY CALCULATION IN ALIGNMENT OF PARALLEL TEXTS AND ITS EVALUATION

Гельбух А.Ф.[(1)]***, Сидоров Г.О.***[(1)]***, Чанона-Эрнандес Л.***[(2)]

[(1)]*Лаборатория естественного языка и обработки текста,*
Центр Компьютерных Исследований (CIC),
Национальный Политехнический Институт (IPN), г. Мехико, Мексика

[(2)]*Инженерный факультет (механика, электричество) (ESIME),*
Национальный Политехнический Институт (IPN), г. Мехико, Мексика

Для пары текстов на разных языках, являющихся переводом друг друга, задача выравнивания состоит в установлении соответствий их структурных элементов (абзацев, предложений, слов). В данной статье описан оптимизационный алгоритм для автоматического выравнивания на уровне абзацев, основанный на вычислении сходства через сравнение их лексического состава, т.е. наличие в одном из текстов словарных переводов слов другого. Представлены результаты экспериментов и сравнение с разными базовыми методами, использующими разные меры вычисления сходства. Эксперименты проводились на литературных текстах, представляющих трудности для выравнивания. Кроме того, предлагается новый способ оценки алгоритмов выравнивания текстов, основанный на восстановлении структуры исходного текста из единиц более низкого уровня: в нашем случае мы восстанавливаем структуру абзацев, формируя их из предложений в одном из текстов. Преимущество такой оценки состоит в том, что устраняется зависимость от обычно тривиальных для абзацев данных существующих корпусов, что дает возможность сравнивать алгоритмы на нетривиальном материале, либо избегается ручная разметка, если использовать для оценки прежнюю структуру абзацев (хотя бы и с некоторой погрешностью), сравнивая ее с полученной.

1. Введение

Для пары текстов на разных языках, являющихся оригиналом и переводом (или же переводами одного и того же оригинала), задача их выравнивания состоит в определении соответствий элементов одного текста элементам другого [10]. Очевидно, что можно выбрать разные структурные элементы для определения таких соответствий: это могут быть абзацы, предложения, и, наконец, слова. В принципе, это могли бы быть и сами тексты – в случае набора коротких текстов, или их крупные структурные единицы вроде глав или частей, но обычно такие задачи не рассматриваются, потому что не очень ясна их практическая ценность. Выравнивание же на уровне предложений и слов является исключительно важным для современной компьютерной лингвистики, так как позволяет автоматически получать ресурсы, используемые как в алгоритмах автоматического обучения, например, в обучении систем статистического машинного перевода, так и в более традиционных задачах, например, в автоматическом составлении двуязычных словарей. Выравнивание на уровне абзацев обычно является первым необходимым шагом перед выравниванием предложений. Заметим, что несмотря на свою полезность, пока не существует достаточного количества параллельных текстов для большинства пар языков, наверное, исключением можно считать пары английский-китайский и английский-арабский.

Выравнивание не является тривиальной задачей, скажем, один абзац может переводится несколькими, какие-то слова могут опускаться при переводе или заменяться выражениями и пр.

Для автоматического выравнивания параллельных тектов использовались различные подходы, обычно на уровне предложений [3][5][18], доступны и различные системы для выравнивания, однако задача точного выравнивания на всех уровнях еще не решена, особенно если рассматривать литературные тексты, где у переводчика есть возможность переводить творчески. Рассматитривая выравнивание на уровне предложений и абзацев с

[1] Work done under partial support of Mexican Government (CONACyT, SNI) and National Polytechnic Institute, Mexico (SIP, COFAA, PIFI).

точки зрения используемых алгоритмов, эти алгоритмы можно разделять на оптимизационные, например, динамическое программирование, и эвристические, например, использование контрольных точек, где перевод очень вероятен, вроде имен собственных или дат. С точки зрения вычисления схожести текстов можно разделять методы, использующие лексическое сходство, и методы, основанные на статистических характеристиках текста, например, на длине выравниваемых единиц или на их положении в тексте. Выравнивание на уровне слов использует несколько другие методы, здесь не рассматириваемые. В нашей предыдущей работе [8], мы представили «жадный» эвристический алгоритм выравнивания на уровне абзацев, использующий технику вычисления лексического сходства для литературных текстов. В данной статье нашей целью является проверить насколько хорошо работает оптимизационный алгоритм с разными мерами вычисления сходства на тех же самых литературных текстах, а также предложить новый способ сравнения алгоритмов и мер сходства, основанный на восстановлении структуры текста.

Работа организана следующим образом. Сначала мы описываем используемые меры выравнивания, которые используются для вычисления функции штрафа/сходства при оптимизации. Затем мы представляем оптимизационный алгоритм выравнивания, основанный на динамическом программировании, находящий наилучшее решение проблемы. После этого, приводятся экспериментальные результаты в соответствии с традиционной схемой оценки и предлагается новый способ оценки. В конце статьи делаются выводы и рассматриваются направления будущих исследований. В приложении приведен код на C++, который реализует алгоритм динамического программирования. Этот алгоритм может быть использован для многих других задач, связанных с выбором оптимального решения, достоточно только изменить функцию, вычисляющую различие (или сходство) двух элементов сравниваемых множеств.

2. Мера сходства

Каждому возможному соответствию между элементами текста (в нашем случае, абзацами) присваивается вес. В предыдущей версии алгоритма мы дополнительно накладывали ограничение на то, что по крайней мере в одном из текстов это должен быть ровно один абзац [9], теперь это ограничение снято, и максимальное количество возможных абзацев в каждом тексте является параметром алгоритма. Для вычисления сходства нескольких абзацев в одном из текстов, эти абзацы виртуально склеиваются и рассматриваются как один абзац.

2.1 Базовые меры сходства, используемые для сравнения

Здравый смысл подсказывает, что соответствующие друг другу абзацы должны находится примерно на одинаковом относительном расстоянии от начала текста. Определим сходство между двумя фрагментами текстов T_A в яыке A и T_B в языке B следующим образом:

$$\text{Distance}(T_A, T_B) = |\text{start}(T_A) - \text{start}(T_B)| + |\text{end}(T_A) - \text{end}(T_B)|,$$

где start(T_X) это относительное положение первого слова текста T_X, вычисленное в процентах от общего числа слов в тексте на соответствующем языке, аналогично для последнего слова end(T_X). В принципе, можно было бы пользоваться относительным положение всего абзаца, но вычисления в «словах» более точные.

В качестве еще одной меры сходства мы воспользовались известным алгоритмом Гейла и Черча [6]. Этот алгоритм учитывает длины предложений, вычисленные в символах и производит дополнительную обрабоку, штрафуя комбинации, отличные от «один к одному».

2.2 Предлагаемая мера сходства

Определим меру сходства между двумя текстами на разных языках как количество слов, которые не являются переводами друг друга [13]. Эта мера сходства является на самом деле мерой различия, то есть штрафом. Это значит, что чем она больше, тем менее похожи тексты. Это удобнее, потому что таким образом косвенно учитывается длина текста, то есть если в тексте очень много непереведенных слов, то, хотя бы и были переведенные, он не очень похож на оригинал. Мы пользуемся термином «сходство» потому что это принятый термин. Следующий алгоритм вычисляет сходство:

1. Установить T_X, := соотвествует короткому тексту T_A или T_B; T_Y := другой текст
2. Количество Переводов := 0
3. для каждого слова w из T_X
4. если какой-нибудь его перевод $D_{XY}(w)$ принадлежит T_Y
5. увеличить Количество Переводов на 1

где $D_{XY}(w)$ это функция возвращающая словарные переводы слова w. Тогда количество слов без перевода в обоих абзацах, предполагая, что они соответствуют друг другу, будет:

$$\text{Сходство}(T_A, T_B) = |T_A| + |T_B| - 2 \times \text{Количество Переводов}.$$

Это стоимость гипотезы о выравнивании двух текстов, соответствующая количеству слов без перевода.

Заметим, что у такого сравнения есть недостатки, по крайней мере на уровне абзацев. Скажем, игнорируется положение слова и его перевода в абзаце. Это можно принять во внимание, например, присваивая веса на основе сходства в относительном положении в абзацах. Правда, это не должно учитываться в слишком коротких абзацах.

Language B

Language A	0	1	2	j	...	N_B
0	0	∞	∞	∞	...	∞
1	∞	0.1	0.3	0.4	0.6	0.8
2	∞	0.3	0.5	0.5	0.7	0.7
3	∞	0.4	0.7	0.7	0.8	0.9
i	...	0.4	0.6	a_{ij}		
N_A	∞					?

Рис. 1. Работа алгоритма динамического программирования.

3. Алгоритм

Для определения наилучшего выравнивания мы применяем алгоритм, основанный на динамическом программировании. Этот алгоритм использует двумерную матрицу $(N_A + 1) \times (N_B + 1)$, показанную на Рис.1, где N_X это количество абзацев на языке X.

Этот алгоритм работает следующим образом. Сначала матрица заполняется наилучшими значениями:

1. $a_{00} := 0$, $a_{i0} := -\infty$, $a_{0j} := -\infty$ для всех $i, j > 0$.
2. for i from 1 to N_A do
3. for j from 1 to N_B do
4. $a_{ij} := \min(a_{xy} + \text{Distance}(T_A[x + 1 .. i], T_B[y + 1 .. j]))$

Здесь a_{ij} это значение ячейки (i,j) матрицы, $T_X[a .. b]$ это набор абзацев от a до b включительно для текста на языке X. Минимум вычисляется для всех ячеек (x,y), находящихся вверху и слева в заданной области для ячейки (i,j). В случае, представленном на Рис.1 в виде ⌟-образной фигуры с тройной границей, реализовано предположение, что соответствие может быть только «один ко многим» или «много к одному», то есть по крайней мере один $T_X[a .. b]$ состоит из одного абзаца. Если мы снимем это условие, то мы должны будем рассматривать всю левую верхнюю часть, или соответствующую прямоугольную подобласть. На Рис.1, приведены случайные значения (0.3, 0.1 и пр.) чтобы показать, что соответствующие ячейки должны быть уже заполнены до вычисления значения в ячейке a_{ij}, см. Приложение, где приводится код алгоритма на C++.

Как и в любом алгоритме динамического программирования, значение a_{ij} это общая цена оптимального выравнивания начальных i абзацев текста на языке A с начальными j абзацами текста на языке B. Это значит, что когда алгоритм заканчивает работу, в ячейке внизу справа, помеченной "?" на Рис.1 содержится общая цена выравнивания двух текстов. Полное выравнивание получается проходом пути, ведущим к этой ячейке:

1. $(i,j) := (N_A, N_B)$.
2. while $(i,j) \neq (0, 0)$ do
3. $(x,y) := \text{argmin}(a_{xy} + \text{Similarity}(T_A[x + 1 .. i], T_B[y + 1 .. j]))$
4. print "абзацы в A с $x + 1$ до i выравнены с
5. абзацами в B с $y + 1$ до j."
6. $(i,j) := (x,y)$

Этот алгоритм распечатает наилучшее выравнивание (в обратном порядке).

4. Экспериментальные результаты: традиционная оценка

Мы провели эксперименты с новеллой *Advances in genetics* (автор Abdón Ubídia) и его испанским оригиналом *De la genética y sus logros*, загруженными с Интернета. В этой новелле перевод является достаточно творческим, см. пример в нашей предыдущей статье [8]. Английский текст содержит 114 абзацев, испанский – 107 абзацев, включая заголовок.[2] Тексты были выравнены вручную, что дало «золотой стандарт» выравнивания. В приведенной ниже таблице приводятся только пары отличные от «один к одному»: то есть, значение 2–3=2 соответствует тому факту, что английские абзацы 2 и 3 соответствуют испанскому абзацу 2. Пары «один к одному» тривиально восстанавливаются из таблицы, например, после 48–50=47 следует 51=48, 52=49, и т. п.

Оба текста были лемматизированы [7], [20] что позволяет правильно определять словарные входы в двуязычных словарях. Служебные слова исключаются из рассмотрения, потому что только добавляют шум. После этого был применен алгоритм выравнивания с различными мерами вычисления сходства. Результаты выравнивания приводятся в Таб. 1.

Мера	Выравнивание
Ручное	2–3=2; 4–5=3–4 (4–5=3, 5=3–4); 6=5–6; 9–10=9; ∅=21; 46–47=46; 48–50=47; 51–53=48; 58–59=53; 87–88=81
Предлагаемое	2–3=2; 4–5=3; 6=4–6; 9–10=9; 22=21–22; 46–47=46; 48–50=47; 51–53=48; 58–59=53; 67=61–62; 68=63–64; 69–71=65; 85=79–80; 86–88=81
Базовое	2–3=2; 4–6=3; 7=4–7; 9–10=9; 11–12=10; 13=11–13; 15–16=15; 22=21–23; 23–24=24; 25–32=25; 33=26–27; 35=29–32; 36–37=33; 38=34–35; 39=36–38; 41=40–41; 42–43=42; 44=43–44; 46–47=46; 48–50=47; 51–53=48; 54=49–50; 55–56=51; 57–58=52; 59–60=53; 61=54–55; 63–64=57; 65=58–60; 66–68=61; 69=62–63; 72–73=66; 76–77=69; 78=70–71; 79=72–73; 82=76–77; 83–84=78; 85=79–80; 86–87=81; 88–89=82; 91–92=84; 94–96=86; 97=87–89; 98=90–91; 99–100=92; 101–102=93; 103–104=94; 105=95–99; 106–108=100; 109=101–102; 111–112=104; 113=105–106
Гейл и Черч	2-3=2; 4-5=3-4; 6=5-6; 7-8=7; 9-10=8-9; 18=17-18; 19-20=19-20; 46-47=46;48-49=47; 50-51=48; 52-53=49; 55-56=51; 58-59=53; 87-88=81;90-91=83-84;

Таб. 1. Сравнение мер вычисления сходства.

Мы оценивали выравнивание, вычисляя точность и полноту для гипердуг графа (гипредуги – это каждое выравнивание (каждое соответствие), взятое само по себе) [11], см. Таб. 2. Точность соответствует совпадению данного выравнивания золотому стандарту (первая строка в Таб. 1), включая пары «один к одному», опущенные в таблице; полнота – это совпадение золотого стандарта с данным выравниванием. Заметим, что это не то же самое, что точность. Дополнительно, мы разбили гипердуги на отдельные элементы, например, 48–50=47 было разбито на 48 ~ 47, 49 ~ 47, 50 ~ 47, и вычислили для них те же самые оценки, см. последнюю колонку в Таб.1.

Мера	Гипердуги		Отдельные дуги	
	Точность, %	Полнота, %	Точность, %	Полнота, %
Предлагаемое	89	85	88	90
Базовое	65	28	43	54
Гейл и Черч	89	86.5	87.5	91.5

Таб. 2. Результаты выравнивания.

Как видно, предлагаемое выравнивание превосходит чисто статистическую меру и находится практически на том же уровне, что и алгоритм Гейла и Черча. Однако, алгоритм Гейла и Черча использует дополнительные заранее вычисленные параметры, то есть не является чисто независимым от обучения (обучение без учителя, unsupervised) алгоритмом. Кроме того, он основан на гипотезе о нормальном распределении. В нашем алгоритме не используются подобные дополнительные данные. В дальнейшем, для практических целей, интересно добавить информацию такого рода в наш алгоритм.

5. Экспериментальные результаты: оценка через восстановление структуры текста

Традиционные методы оценки [3] обычно включают прямое сравнение с золотым стандартом, см. также формализацию подобного сравнения в [12]. Могут вычислятся как точность и полнота, так и их совместная мера F1. Заметим, что в той же статье упоминается, что эти значения могут вычисляться, используя единицы более

[2] Мы не провели более масштабных экспериментов, так как нам неизвестен выравненный вручную соответствующий корпус литературных тектстов.

низких уровней, т. е., скажем, на уровне предложений могут подсчитываться правильно выравненные слова или символы. В указанной работе выравнивание на уровне абзацев не упоминается.

Таким образом, при оценке у нас есть две возможности:

- Сравнивать с золотым стандартом. В этом случае мы считаем соответствия элементов того же уровня, то есть находимся на том же самом уровне.
- Сравнивать правильные соответствия единиц более низкого уровня. Скажем, в случае абзацев, считать соответствия слов или предложений.

Приняв второй подход для нашей проблемы выравнивания на уровне абзацев, мы можем переформулировать задачу как восстановление глобальной структуры текста на уровне абзацев. То есть, сначала мы уберем все границы абзацев в одном из текстов, будем считать абзацем каждое предложение, а затем попытаемся расставить границы абзацев на основе данных из другого текста, выравнивая предложения (=абзац) с абзацами на другом языке. Затем можно применить традиционную оценку выравнивания на уровне абзацев, на основе данных о сохраненных абзацах другого языка. В этом случае мы не может считать, что мы восстановим правильно всю исходную структуру, потому что мы привлекли данные из другого языка, а само выравнивание исходно не симметрично: может быть 2 к 1, 3 к 1, и пр. Если мы готовы примириться с обычно небольшим процентом таких ошибок, то это избавляет нас от необходимости вручную выравнивать тексты для последующей оценки.

Восстановление структуры текста похоже на оценку с использованием единиц нижних уровней, но только в том смысле, что рассматриваются единицы таких уровней. Основное отличие состоит в том, что когда алгоритм восстанавливает структуру текста, он принимает во внимание возможные комбинации, которые не существуют в реальных текстах. Это особенно хорошо видно на примере выравнивания на уровне абзацев.

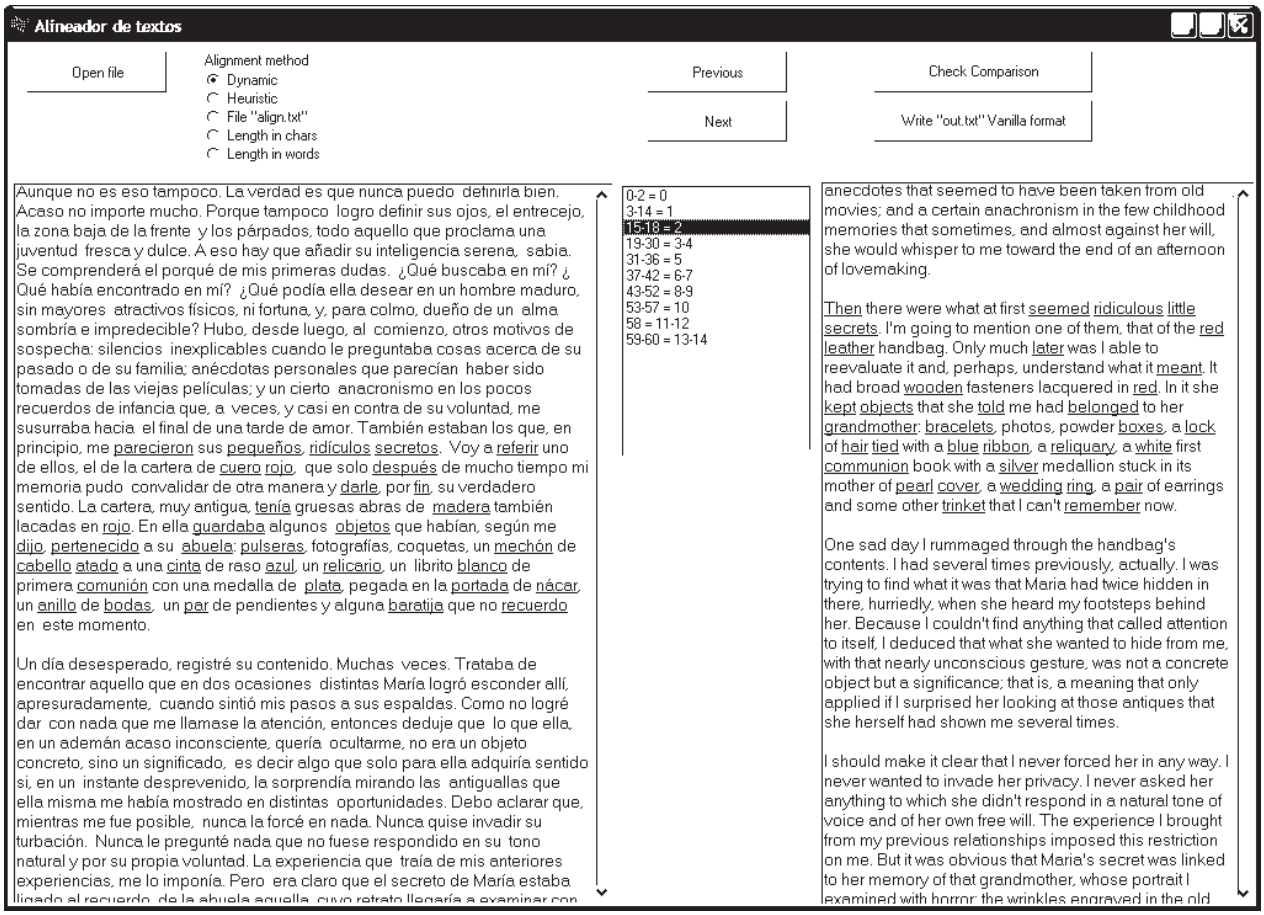

Рис. 2. Система выравнивания текстов

Обычно выравнивание на уровне абзацев не рассматриваетя как интересная проблема, потому что в большинстве доступных корпусов параллельных текстов, даже в очень больших корпусах, абзацы обычно выравнены один к одному. Поэтому традиционная оценка выравнивания абзацев на существующих корпусах обычно тривиальна. Но теперь, если мы будем рассматривать задачу как восстановление структуры текста, задача выравни-

вания на уровне абзацев становится интересной и нетривиальной проблемой. Дополнительно, у нас появляется возможность сравнивать разные подходы к выравниванию на уровне абзацев на нетривиальных данных. Заметим, что мы не пытаемся создать проблему там, где ее реально не существует. То, что мы утверждаем, что если в нашем корпусе нет трудных случаев, это не значит, что они никогда не появятся, и поэтому надо заранее оценить, как они будут обработаны.

Напомним, что существующие параллельные корпуса обычно состоят из юридических или технических текстов, где переводческие вольности не поощряются, а ценится наибольшее структурное соответствие оригиналу, поэтому на уровне абзацев оригинал и перевод различаются редко. Применяя предлагаемый способ выравнивания через восстановление структуры текста, мы решаем проблему нехватки данных для оценки такого выравнивания, потому что переходим к единицам более низкого уровня. Это позволяет применять и сравнивать различные меры вычисления сходства или алгоритмы выравнивания.

Мы провели эксперимент оценки выравнивания способом восстановления струкутуры текста, используя описанный выше алгоритм динамического программирования. Нашей задачей было сравнить, используя этот способ оценки, статистический подход к вычислению сходства при выравнивании – мы взяли алгоритм Гейла и Черча [6], основанный на длинах абзацев – и лексический подход.

В эксперименте мы рассматривали каждое предложение одного из текстов как отдельный абзац и пытались выравнять их с реальными абзацами другого текста. При этом размер окна в алгоритме динамического программирования должен быть увеличен до максимального числа предложений в абзаце, или даже, для более объективной оценки, быть несколько больше.

При использовании алгоритма Гейла и Черча при этом способе возникает техническая проблема, связанная с его текущей реализацией. Максимально возможное количество элементов выравнивания в нем равно двум, то есть максимальное соответствие это 2-2 (т.е., скажем, 3-2 уже невозможно). Кроме того, в нем присваиваются эмирические значения штрафов для соответствий отличных от 1-1. Мы попытались экстраполировать эти штрафы на другие соответствия, например, 10-1, и т.п. Очевидно, что качество экстраполяции может повлиять на работу алгоритма. В дальнейшем необходимо исследовать, насколько удачна наша экстраполяция и какие еще изменения надо внести в стандартный алгоритм Гейла и Черча, чтобы его применение с новым способом оценки было полностью корректным.

В качестве лексического подхода мы использовали нашу меру вычисления сходства, описанную выше. Пока мы не добавляли в нашу меру штрафов за разные размеры абзацев, за разное положение абзацев в тексте или за разное положение слов и их переводов в абзацах при их сравнении. Мы ожидаем, что эти дополнения улучшат работу нашего алгоритма. Практически это значит, что алгоритм с такими изменениями будет комбинированным алгоритмом, одновременно использующим обе меры вычисления сходства. Обычно на практике, комбинированные подходы работают лучше, чем каждый из них в отдельности.

Мы провели эксперимент на выборке из 15 абзацев из текста, упомянутого выше. Напомним, что этот текст является трудным случаем с достаточно свободным переводом. Мы применили алгоритм динамического программирования, выравнивая предложения на одном языке (испанском) с абзацами на другом (английском), то есть восстанавливая структуру испанского текста.

Результаты эксперимента таковы: точность равна 84% при лексическом подходе и только 26% при статистическом подходе. Мы подсчитывали точность на основе структуры абзацев английского текста. В случае, когда алгоритм объединял два абзаца, которые были разделены в обоих текстах, мы считали это ошибкой для половины выравненных с таким абзацем предложений, и считали это правильным для другой половины. В будущем интересно исследовать вопрос, является ли этот тип ошибок таким же, как просто неправильное выравнивание.

Одна из проблем статистического алгоритма состояла в том, что один раз сбившись, он достаточно долго продолжал выравнивать неправильно.

На Рис.2 приведен экран системы выравнивания текстов. Слева приведен текст на испанском языке, справа – на английском языке, в середине приведены соответствия между структурными элементами текстов. Подчеркнутые слова имеют перевод в тексте на другом языке. Структурные элементы испанского текста соответствуют предложениям, и им поставлены в соответствие абзацы английского текста, т.е. структура текста на испанском языке восстановлена на основе структуры английского текста.

6. Выводы

В данной работе описан алгоритм выравнивания параллельных текстов, относящийся к классу методов обучения без учителя (unsupervised learning). Метод основан на сходстве лексического состава текстов. Для поиска оптимального выравнивания используется алгоритм динамического программирования. Эксперименты по сравнению различных мер вычисления сходства проводились на англо-испанском корпусе художественных тек-

стов, отличающемся очень вольным переводом, сильно затрудняющим выравнивание. Проведенные эксперименты показали, что качество полученного выравнивания очень близко к качеству выравнивания с помощью алгоритма Гейла-Черча, в то время как этот последний алгоритм относится к классу методов обучения с учителем (supervised learning).

Кроме того, в работе предложен новый метод оценки качества выравнивания параллельных текстов. Метод заключается в попытке восстановления структуры текста на одном из двух языков с использованием единиц более низкого уровня в тексте на другом из языков. Например, в случае выравнивания на уровне абзацев, абзацы текста на одном из языков автоматически формируются из предложений с использованием информации о предложениях и абзацах в тексте на другом языке, и результат сравнивается с ручным выравниваем. Как вариант, с исходной структурой текста, но при этом заранее известно, что возможен какой-то процент ошибок, связанный с исходной несимметричностью выравнивания.

Преимущество такого метода оценки заключается в том, что он не требует реального корпуса с нетривиальным соответствием абзацев, либо размеченного вручную, если рассматривать исходную структуру абзацев. В случае задачи выравнивания на уровне абзацев, оценка качества алгоритмов выравнивания затруднена тем, что все алгоритмы имеют очень высокую точность: почти всегда определить соответствие между абзацами легко, и разница в качестве работы алгоритмов проявляется только в редких случаях. Для нахождения таких нетривиальных случаев выравнивания нужен очень большой корпус, поскольку в маленьком корпусе почти все абзацы выравнены тривиально. Однако ручная разметка такого корпуса требует больших затрат труда, поскольку для этого все равно требуется прочитать весь корпус.

Был проведен эксперимент, где качество выравнивания определялось с помощью предложенного метода оценки. Были проведены эксперименты по выравниванию с использованием лексического и статистического подхода к определению меры близости между абзацами. В обоих случаях оптимальное (для данной меры близости) выравнивание определялось с помощью алгоритма динамического программирования. Намного лучшие результаты был получены с помощью предложенного лексического метода, хотя статистический метод явно может быть улучшен в приложении к данной задаче.

Среди будущих направлений исследования могут быть упомянуты следующие:

– Анализ ошибок: необходимо проанализировать типичные причины ошибок, совершаемых предложенным алгоритмом. На самом деле только такой анализ поможет действительно определить направления дальнейших исследований.
– Улучшение алгоритма определения меры близости между абзацами. Например, в нынешней реализации алгоритм может переводы некоторых слов посчитать дважды (например: *он подобрал ключ к этому потоку данных* / *he deciphered the data stream*). Впрочем, решение этой сравнительно редко проявляющейся проблемы привело бы к гораздо большей сложности алгоритма – практически для этого потребовалось бы выполнить выравнивание на уровне отдельных слов.
– Разработка меры близости между абзацами с учетом порядка слов.
– Применение весов слов типа TF-IDF. Имеющаяся реализация просто удаляет стоп-слова, а остальные слова считает с одинаковым весом.

Список литературы

1. Brown, P. F., Lai, J. C. & Mercer, R. L. 1991. Aligning Sentences in Parallel Corpora. // In: Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics, Berkeley, California, pp. 169–176.
2. Baeza-Yates, R., B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, 1999.
3. Caseli, H. M., and M. G. Volpe Nunes. 2003. Evaluation of Sentence Alignment Methods on Portuguese-English Parallel Texts. // Scientia 14(2):1–14.
4. Chen, S. 1993. Aligning sentences in bilingual corpora using lexical information. // In: Proceeding of ACL-93, pp. 9–16.
5. Bing Zhao et al. 2003. Efficient Optimization for Bilingual Sentence Alignment based on Linear Regression. // In: HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond. p. 81–87.
6. Gale, W. A. & Church, K. W. 1991. A program for Aligning Sentences in Bilingual Corpora. // In: Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics, Berkeley, California.
7. Gelbukh, Alexander, and Grigori Sidorov. 2003. Approach to construction of automatic morphological analysis systems for inflective languages with little effort. // Lecture Notes in Computer Science, N 2588, Springer-Verlag, pp. 215–220.

8. Гельбух А.Ф., Сидоров Г.О., Вера-Феликс А. Словари в задачах автоматической обработки пар переводных текстов. // Труды межд. конф. Диалог-2006, М.: 2006. С. 110-114
9. Alexander Gelbukh and Grigori Sidorov. 2006. Alignment of Paragraphs in Bilingual Texts using Bilingual Dictionaries and Dynamic Programming. // Lecture Notes in Computer Science, N 4225, Springer-Verlag, 2006, pp 824-833.
10. Kay, Martin and Martin Roscheisen. 1993. Text-translation alignment. // Computational Linguistics, 19(1):121–142.
11. Kit, Chunyu, Jonathan J. Webster, King Kui Sin, Haihua Pan, Heng Li. 2004. Clause alignment for Hong Kong legal texts: A lexical-based approach. // International Journal of Corpus Linguistics 9:1. pp. 29–51.
12. Langlais, Ph., M. Simard, J. Veronis. 1998. Methods and practical issues in evaluation alignment techniques. // In: Proceeding of Coling-ACL-98.
13. McEnery, A. M. & Oakes, M. P. 1996. Sentence and word alignment in the CRATER project. // In: Using Corpora for Language Research, London, pp. 211–231.
14. Melamed, I. Dan. 1996. A Geometric Approach to Mapping Bitext Correspondence. // Proc. EMNLP-1996, ACL, pp. 1–12.
15. Melamed, I. Dan. 2000. Pattern Recognition for Mapping Bitext Correspondence. // In: Parallel Text Processing: Alignment and Use of Translation Corpora. Kluwer, pp. 25–47.
16. Meyers, Adam, Michiko Kosaka, and Ralph Grishman. 1998. A Multilingual Procedure for Dictionary-Based Sentence Alignment. // In: Proceedings of AMTA'98: Machine Translation and the Information Soup, pp. 187–198.
17. Mikhailov, M. 2001. Two Approaches to Automated Text Aligning of Parallel Fiction Texts. // Across Languages and Cultures, 2:1, pp. 87–96.
18. Robert C. Moore. 2002, Fast and Accurate Sentence Alignment of Bilingual Corpora. // In Proc. AMTA-2002. pp. 135–144.
19. Simard, M., George Foster, Pierre Isabelle. 1992. Using Cognates to Align Sentences in Bilingual Corpora. // In: TMI-1992, pp. 67–81.
20. Velásquez, F., Gelbukh, A. & Sidorov, G. 2002. AGME: un sistema de análisis y generación de la morfología del español. // In: Proc. of Workshop on Multilingual information access & natural language processing of IBERAMIA 2002, pp 1–6.

Приложение. Код на C++, реализующий алгоритм динамического программирования

Ниже приводится код, реализующий описанный алгоритм динамического программирования. Этот код полностью функционален. Как видно, он не очень сложен. Мы полагаем, что код может быть полезен читателю для решения его собственных оптимизационных задач. Код позволяет найти оптимальный путь при выравнивании двух множеств. Читатель должен определить функцию Similarity(), возвращающую сходство пары элементов этих множеств.

Заметим, что наша матрица логически начинается с -1, поэтому везде в адресации появляется +1 для приведения к 0 значений индексов.

Структура данных vector реализует динамический массив и доступна в библиотеке STL.

```
struct Chart
{
	Chart ()  :   val (FLT_MAX), i (-1), j (-1)  {   }

	void SetIfBetter (Chart &v, int i, int j, int i0, int j0);

	float val;
	int i;
	int j;
};

void Chart::SetIfBetter (Chart &v, int this_i, int this_j, int i0, int j0)
{
	float sim = Difference (i0 + 1, this_i, j0 + 1, this_j) ;

	float new_val = v.val + sim;
```

```
        if (new_val < val)   // val — это текущее значение в данном элементе Chart
        {                    //  Если Difference будет вычислять сходство, а не различие, то  «>»
                             // знак «<» заменяется на и везде вместо FLT_MAX ставится  -FLT_MAX
                val = new_val;
                i = i0;
                j = j0;
        }
}

float Difference (int i0, int i, int j0, int j)
{
        return (Здесь добавляется код, определяющий, насколько различны элементы (i,j) и (i0, j0));
}

void Align (int a, int b, int max_match_i, int max_match_j )
{
        // Метод динамического программирования

        vector < vector <Chart> > c (a+1); // Начальные значения

        vector <Chart> v (b+1);

        for (int i = -1; i < a; i++)
                c [i+1] = v;

        c [-1+1][-1+1].val = 0;

         // Параметры max_match_i и max_match_j определяют размер сравниваемой области

         for (int i = 0; i < a; i++)          // Заполнение
        {
                for (int j = 0; j < b; j++)
                {
                        for (int i0 = max (-1, i - max_match_i); i0 < i; i0++)
                          for (int j0 = max (-1, i - max_match_j); j0 < j; j0++)
                          {
                                c [i+1][j+1].SetIfBetter (c [i0+1][j0+1], i,  j, i0, j0);
                          }
                }
        }

        int i = a - 1;
        int j = b - 1;

        while (i != -1 || j != -1)           // Распечатка результатов, в обратном порядке
        {
                Chart & cur = c [i+1][j+1];

                cout << i  << "=" << j << endl;

                i = cur.i;
                j = cur.j;
        }
}
```