Параллельная реализация алгоритма обучения системы текстовой классификации

Т.А. Пескишева, Е.В. Котельников

Вятский государственный гуманитарный университет

В статье рассматривается алгоритм параллельного обучения системы текстовой классификации на основе метода опорных векторов (SVM) и стратегии многоклассовой классификации «один против всех». Предложено три метода распределения нагрузки между узлами вычислительного кластера. Для каждого из методов приводятся результаты экспериментов на текстовой коллекции Reuters-21578 и характеристики производительности.

1. Введение

Текстовая классификация (рубрикация) – отнесение текстовых документов к одной или нескольким заранее заданным категориям (рубрикам) в соответствии с определенными признаками. В условиях экспоненциального роста объема накапливаемой и используемой человечеством текстовой информации, хранящейся в электронном виде, задача тематической текстовой рубрикации становится все более актуальной, но вместе с тем сложной и трудоемкой.

На сегодняшний день для решения данной задачи используется множество различных программных продуктов. Существующие системы и модули текстовой классификации демонстрируют приемлемую скорость и точность обработки лишь на небольших и средних по объему данных. В случае значительного роста объема обрабатываемой информации, а также увеличения числа рубрик, по которым необходимо классифицировать документы, их производительность существенно снижается [1].

Решением данной проблемы может быть применение высокопроизводительных методов машинного обучения в реализации системы текстовой классификации. Одним из них является метод опорных векторов (Support Vector Machines, SVM), предложенный В. Н. Вапником [2].

Метод опорных векторов был разработан для решения задач распознавания образов двух классов. Применение SVM сводится к двум основным процессам: обучению и классификации (или распознаванию). Наиболее трудоемким является процесс обучения, который происходит на основе множества векторов. Каждый вектор представляет собой набор признаков, характеризующих распознаваемый объект. В задаче тематической текстовой классификации элементами векторов будут веса слов, входящих в определенный текстовый документ [3]. Вес дает числовую оценку значимости данного слова для определения тематики текста. Каждому вектору сопоставлено число, обозначающее класс, к которому относится вектор.

В процессе обучения в n-мерном пространстве признаков строится линейная или нелинейная гиперплоскость, разделяющая векторы разных классов. Обучение сводится к решению задачи квадратичной оптимизации с предельными ограничивающими условиями и одним ограничением линейного равенства. При этом SVM выделяет так называемые *опорные векторы* – такие векторы, которые находятся ближе всего к разделяющей гиперплоскости. Только опорные векторы несут всю информацию о разделении классов, так что остальные векторы могут в дальнейшем не учитываться.

Особенностями задачи текстовой классификации является, во-первых, очень высокая размерность пространства признаков – десятки, иногда сотни тысяч, во-вторых, большое количество классов (рубрик) – от нескольких десятков до сотен.

Проблема использования SVM на реальных коллекциях текстов связана как с указанными особенностями задачи текстовой классификации, так и с тем, что методы решения задачи квадратичной оптимизации известны, но вычислительно сложны.

В результате время обучения рубрикатора оказывается неприемлемо длительным. Повышение скорости обучения на основе SVM возможно за счет использования многопроцессорных вычислительных систем и комплексов.

Цель данной работы – описать параллельный алгоритм обучения рубрикатора системы многоклассовой текстовой классификации на основе метода опорных векторов и проанализировать эффективность методов распределения нагрузки между узлами вычислительного кластера.

Статья построена следующим образом. Раздел 2 посвящен общим подходам к решению задачи многоклассовой классификации в SVM. В разделе 3 предлагаются три метода распределения нагрузки между узлами кластерной системы при параллельном обучении SVM. В разделе 4 рассматриваются программные и аппаратные аспекты проведенных экспериментов, а также описывается коллекция обучающих текстов Reuters-21578. В разделе 5 приводятся и анализируются результаты экспериментов, а также вычисляются характеристики производительности параллельных методов. Раздел 6 является заключительным и содержит общие выводы по работе и рекомендации.

2. Стратегии многоклассовой классификации в SVM

Существуют два основных подхода к решению проблемы многоклассовой классификации в SVM. Первый называется *«решение за один шаг»* или *«все вместе» («all-together»)* [4]. В этом подходе задача квадратичной оптимизации усложняется за счет увеличения размерности дополнительных переменных.

В другом подходе решение задачи многоклассовой классификации сводится к решению последовательности задач с двумя классами. При этом может быть несколько стратегий генерации такой последовательности: «один против всех» («one-against-all»), «каждый против каждого» («one-against-one»), «турнир на выбывание» или «ориентированный ациклический граф SVM» (Directed Acyclic Graph SVM, DAGSVM).

В стратегии *«один против всех»* [5] для N классов обучается N классификаторов, каждый из которых отделяет «свой» класс от всех остальных классов. На этапе распознавания неизвестный вектор X подается на все N классификаторов. Принадлежность вектора X определяется тем классификатором, который выдал наибольшую оценку $f(X)$.

Стратегия *«каждый против каждого»* [6] выделяет $N(N–1)/2$ классификаторов, обучающихся отличать все возможные пары классов друг от друга. Для распознаваемого вектора каждый классификатор выдает оценку $f_{ij}(X)$, отражающую принадлежность к классам i и j. Результатом является класс с максимальной суммой

$$\sum_{i \neq j} g\left(f_{ij}(X)\right), \tag{1}$$

где g – монотонно неубывающая функция, например тождественная или логистическая.

Стратегия *«турнир на выбывание»* [7] также предполагает обучение $N(N–1)/2$ классификаторов, различающих все возможные пары классов. В отличие от предыдущей стратегии, на этапе классификации вектора X между классами устраивается турнир. При этом на каждом шаге распознавание вектора X осуществляет только один классификатор – «победивший» класс продолжает борьбу и определяет следующий используемый классификатор. Процесс осуществляется до тех пор, пока не останется один победивший класс, который и будет считаться результатом распознавания.

Методы сведения многоклассовой классификации к бинарной обучаются быстрее и дают меньшее число ошибок, в то время как в подходе «решение за один шаг» получается меньшее число опорных векторов.

В [8] подробно описаны параллельные реализации стратегий сведения многоклассовой классификации к бинарной, а также результаты экспериментов с ними. Однако, несмотря на то, что все предложенные способы распараллеливания продемонстрировали свою эффективность, для дальнейшего использования в системе текстовой классификации была выбрана стратегия «один против всех». Выбор именно этой стратегии связан с тем, что она позволяет относить каждый классифицируемый объект сразу к нескольким классам. В задачах текстовой класси-

фикации это существенно, так как часто возникают ситуации, когда документ может быть отнесен сразу к нескольким темам.

3. Подходы к распределению нагрузки

В данном разделе предлагается три метода распределения нагрузки на узлы кластера при параллельном обучении классификатора на основе метода опорных векторов. Во всех подходах используется стратегия сведения многоклассовой классификации к бинарной «один против всех».

Все предлагаемые подходы используют принцип «master-slave». Данный принцип состоит в следующем. Главный узел («master») передает подчиненным узлам («slave») задания, которые они должны выполнить. Подчиненные узлы независимо друг от друга выполняют свои задания, затем полученные результаты возвращают главному узлу.

Методы распределения нагрузки разделены на две группы в зависимости от того, в какой момент времени принимается решение о передаче обучающих векторов на узлы: заранее для примеров всех классов на основе имеющейся информации о векторах (статическое распределение) или в процессе обучения, для каждого класса отдельно (динамическое распределение).

3.1. Статическое распределение на основе примеров

В этом подходе главный узел распределяет примеры всех классов по узлам предварительно, до стадии обучения. Распределение осуществляется на основе известной информации относительно примеров.

Для равномерной загрузки вычислительных узлов желательно иметь информацию о времени обучения для каждого класса. Очевидно, до завершения процесса обучения эти данные недоступны, поэтому в первом методе (**статическое распределение на основе примеров**) для распределения нагрузки используется информация о количестве обучающих примеров в каждом классе и «жадный алгоритм». В общем случае работа «жадного» алгоритм («Greedy algorithm») заключается в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным. На практике при решении задачи сбалансированного распределения нагрузки «жадный» алгоритм почти всегда дает решение в достаточной степени близкое к оптимальному.

В предлагаемом подходе жадный алгоритм применяется следующим образом. Сначала главный узел сортирует классы по убыванию количества примеров. Каждому подчиненному узлу выделяется по одному классу в порядке невозрастания количества примеров. После того, как были распределены первые N классов, остальные классы распределяются следующим образом. Очередной класс выделяется тому узлу, у которого общее количество всех выделенных ему примеров минимально (независимо от того, сколько классов уже выделено данному узлу).

При этом для каждого подчиненного узла формируется и отправляется собственный список рубрик, подлежащих обработке. На основе этого списка отбирается и пересылается набор обучающих векторов для каждого подчиненного узла. Узел, получив эти данные, строит правило классификации (модель) для каждой переданной ему рубрики. Завершив обучение, узел передает построенные модели на главный узел. Работа кластера завершается, когда последний узел закончит вычисления и передаст результаты главному узлу.

Данный подход показан на рис. 1, операции, выполняемые только главным узлом, обозначены буквой «M», а подчиненными узлами – буквой «S».

В рассматриваемом методе реальная загрузка узлов оказывается иногда далека от оптимальной по той причине, что время обучения SVM зависит не только от количества обучающих векторов, но и от их взаимного расположения в n-мерном пространстве. Для обеспечения большей сбалансированности требуется знать время обучения (или количество опорных векторов) для каждого класса.

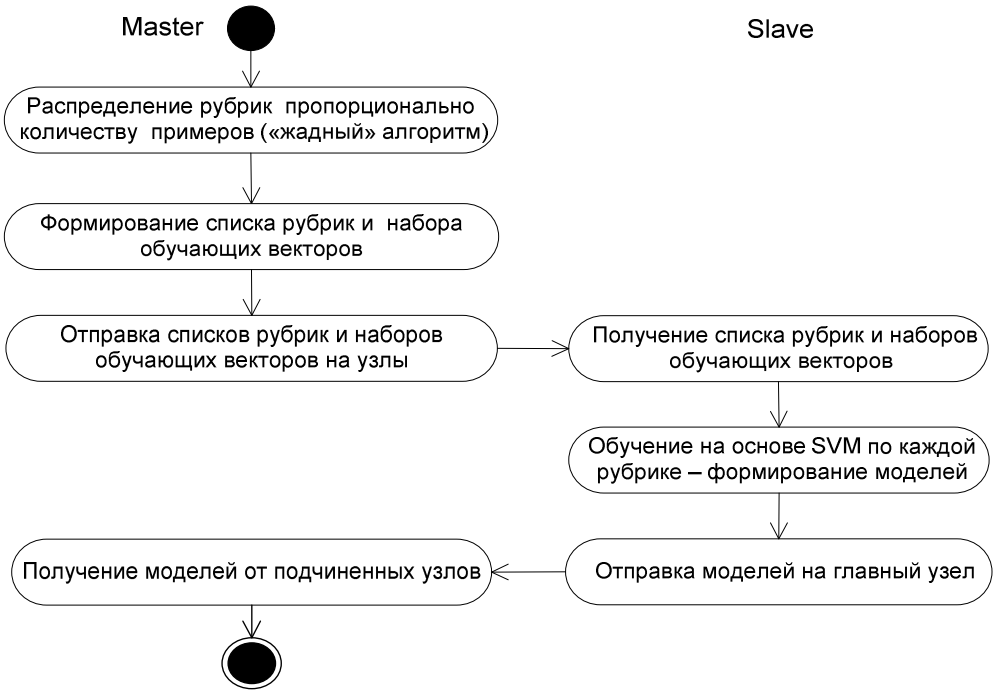

Рис. 1. Выполнение параллельного алгоритма при статическом распределении на основе примеров

3.2. Статическое распределение на основе опорных векторов

В некоторых ситуациях может требоваться многократное обучение классификатора на основе SVM на одинаковых данных, например, при поиске оптимальных параметров. В этом случае время обучения (и опорные векторы) становится известно после выполнения первого прохода и для остальных проходов эта информация может использоваться для распределения рубрик по вычислительным узлам. Предлагается учитывать не время обучения, поскольку оно зависит от аппаратных характеристик узлов, состояния сети и других случайных факторов, а количество опорных векторов, которое является объективной характеристикой набора обучающих векторов для данного алгоритма и параметров обучения. Время обучения классификатора находится в прямой зависимости от количества опорных векторов. Чем больше опорных векторов в классе, тем больше времени потребуется для построения классификатора.

Во втором методе (**статическое распределение на основе опорных векторов**) главный процесс также выполняет предварительное распределение классов по узлам на основе «жадного» алгоритма. Отличие от первого метода в том, что классы сортируются не по количеству примеров, а по количеству опорных векторов. В результате получается более сбалансированная нагрузка на узлы и уменьшается время простоя узлов.

3.3. Динамическое распределение

Основным недостатком первого метода является отсутствие в реальных задачах достаточной сбалансированности нагрузки на узлы вычислительной системы: некоторые узлы намного раньше завершают вычисления и простаивают, как результат – неоптимальное использование ресурсов кластера. Во втором методе этот недостаток отсутствует, но область его применения ограничена задачами поиска оптимальных параметров, в общем случае метод не применим.

Идея третьего метода (**динамическое распределение**) заключается в том, что узлы загружаются по мере их освобождения (динамически). Схема данного метода показана на рис. 2.

Прежде всего, главный узел отправляет все обучающие векторы на каждый подчиненный узел и формирует список свободных узлов. Подчиненные узлы сообщают о своей готовности и в ответ получают информацию о том, какую часть набора обучающих данных должен обработать каждый из них. По окончании вычислений подчиненный узел пересылает результат главному узлу и ждет сообщение с указаниями о необходимости обработки очередной части набора

данных, либо команду завершения. Главный узел выделяет подчиненному узлу следующую порцию обучающих векторов.

Данный подход не использует априорную информацию о количестве обучающих примеров или опорных векторов, т.е. является универсальным, однако при этом увеличивается количество обменов информацией между главным и подчиненными узлами кластера и время ожидания узлов по сравнению со статическими методами, что в некоторой степени уменьшает эффективность алгоритма.

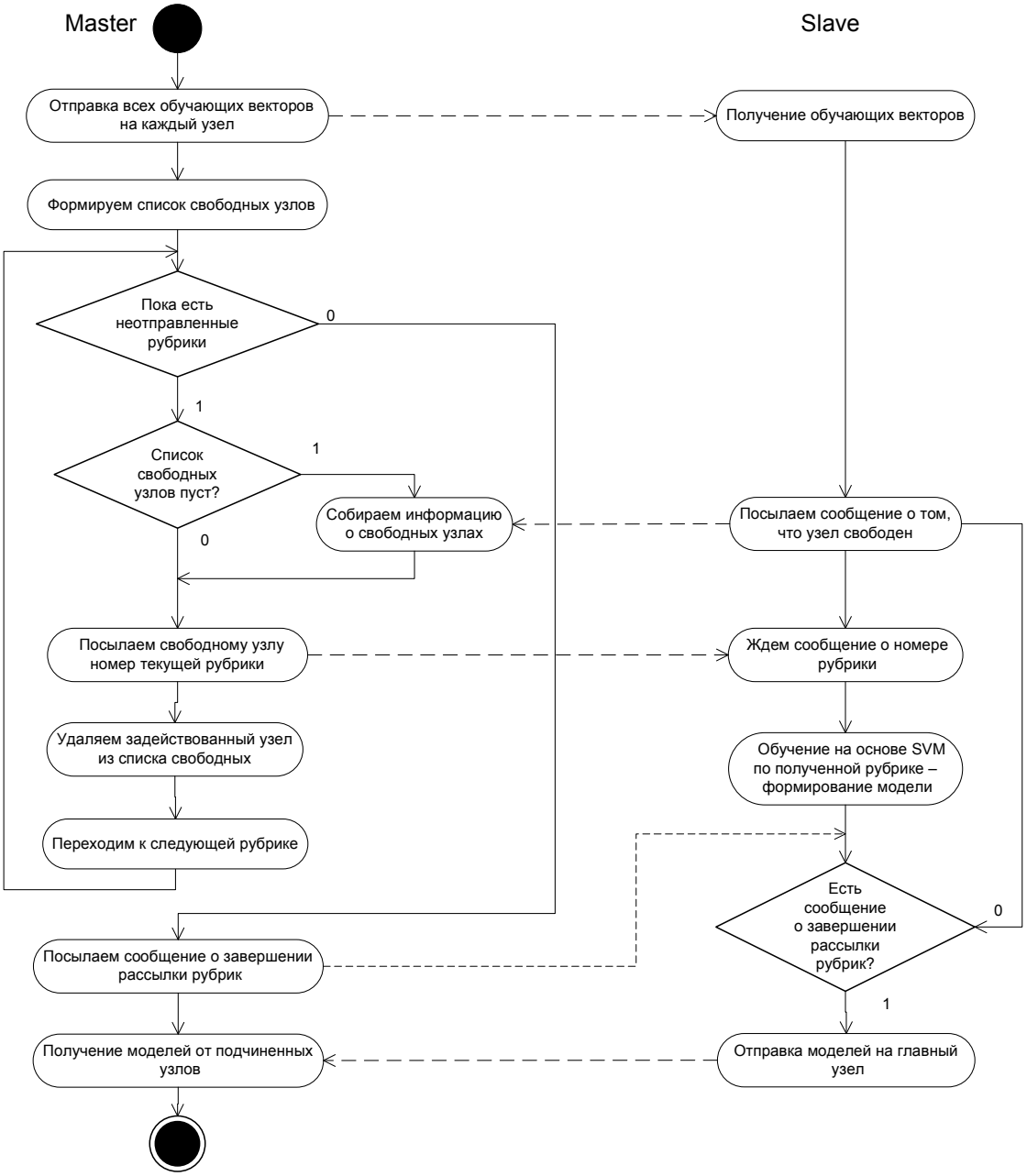

Рис. 2. Выполнение параллельного алгоритма при динамическом распределении нагрузки

4. Условия проведения экспериментов

Для тестирования и анализа предложенных методов распределения нагрузки в параллельном алгоритме обучения многоклассовой классификации было разработано программное при-

ложение для кластерной архитектуры. Приложение создавалось в среде Microsoft Visual Studio 2008 на языке C# на основе принципов объектно-ориентированного программирования.

Для кластерной архитектуры параллельная реализация алгоритмов обучения проводилась с использованием библиотеки MPI.NET версии 1.0 [9]. Библиотека MPI.NET представляет собой свободно доступную реализацию интерфейса передачи сообщений MPI для среды Microsoft.NET и позволяет разрабатывать приложения для MPI на C# и других языках .NET.

Расчеты проводились на вычислительном кластере Вятского государственного гуманитарного университета, состоящем из 30 вычислительных узлов. Каждый вычислительный узел представляет собой персональный компьютер с процессором Intel Core 2 Duo 2 ГГц и 2 Гб оперативной памяти. Узлы связаны сетью Gigabit Ethernet. Кластер функционирует на базе операционной системы Microsoft Windows HPC Server 2008, на каждом узле установлена среда выполнения MPI.NET Runtime версии 1.0.

Для экспериментального исследования использовалась коллекция финансовых новостей агентства Reuters (Reuters-21578, Distribution 1.0) [10]. Обучающий набор документов был выделен в соответствии с общепринятым подходом, названным «ModApte split» [10]. При этом исключались документы, не помеченные ни одной рубрикой. Таким образом, обучающий набор в наших экспериментах представлен 7775 документами, каждый из которых относится к одной или нескольким из 115 рубрик.

Для формирования векторов использовался морфологический анализатор Mystem 1.0 от компании Yandex [11], терминами считались все слова в нормальной форме, исключая стоп-слова. Для взвешивания терминов и получения числовых значений компонентов векторов применялся метод TF.IDF [12]. В результате получены 7775 обучающих векторов, размерность которых составляет 6103.

5. Результаты экспериментов

В ходе экспериментов тестировались три предложенных метода распределения нагрузки между узлами вычислительного кластера. Количество узлов для каждого метода менялось от 3 до 29 (главный узел здесь учитывается, хотя непосредственно для обучения SVM не используется, а играет роль диспетчера), проводилось 5 запусков одного эксперимента с фиксированными параметрами, затем результаты усреднялись.

Для вычисления характеристик производительности параллельных алгоритмов было измерено время выполнения последовательной версии решения данной задачи на одном узле кластера. Полученное значение составляет 68,7 с.

Результаты экспериментов приведены на рис. 3.

Вычислим основные характеристики производительности – ускорение и эффективность.

Ускорение вычисляется как отношение времени решения задачи на одном вычислительном узле T_s ко времени решения на p идентичных вычислительных узлах T_p:

$$S = \frac{T_s}{T_p}. \tag{2}$$

Эффективность отражает долю времени выполнения алгоритма, в течение которой вычислительные узлы были реально задействованы для решения задачи:

$$E = \frac{S}{p}. \tag{3}$$

Значения ускорения и эффективности приведены на рис. 4, 5.

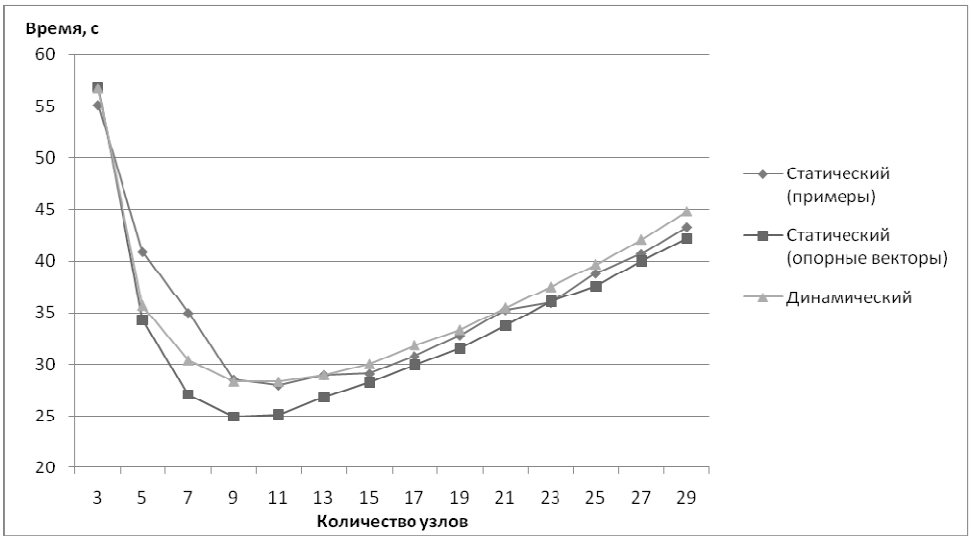

Рис. 3. Результаты экспериментов для трех методов

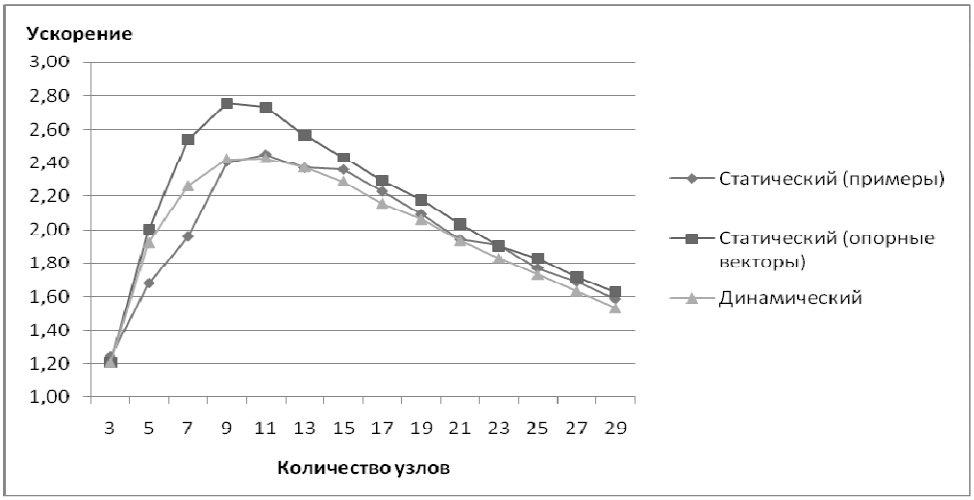

Рис. 4. Зависимость ускорения от количества узлов

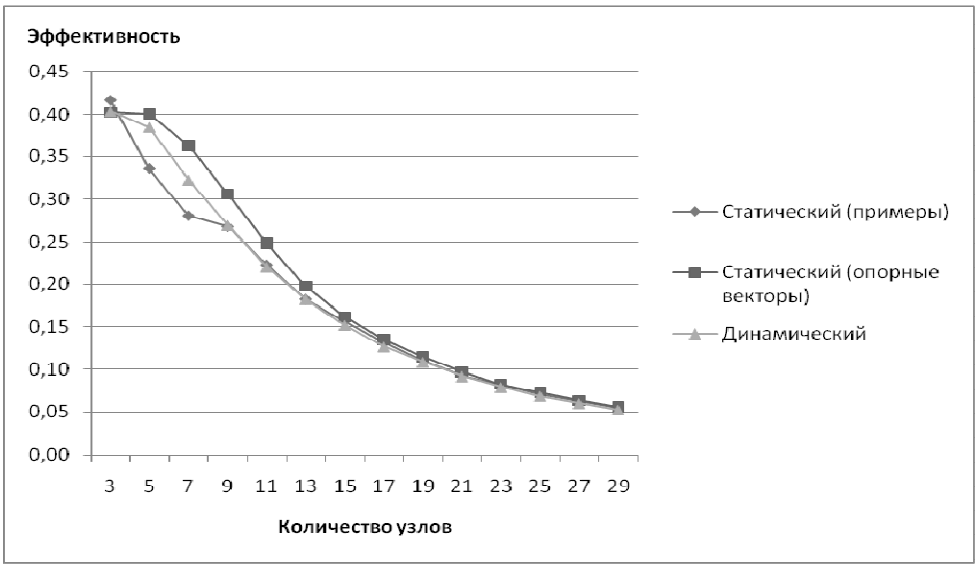

Рис. 5. Зависимость эффективности от количества узлов

По результатам экспериментов можно сделать следующие выводы.

1. Максимальное ускорение во всех методах достигается при количестве узлов 9–11, с увеличением числа узлов наблюдается ухудшение. Связано это с тем, что при количестве узлов больше 11 (для данной коллекции) время, затраченное на ожидание и пересылку информации о текущих рубриках, оказывается больше, чем эффект за счет распараллеливания. Для других обучающих коллекций большего размера точка минимума будет сдвигаться в сторону увеличения количества узлов.

2. При увеличении числа узлов эффективность падает в связи с увеличением времени ожидания каждым узлом списка рубрик (при высокой латентности коммуникационной сети).

3. Наибольшее значение ускорения (2,75 для 9 узлов), как и наибольшие средние значения ускорения и эффективности достигаются в методе статического распределения на основе опорных векторов. В то же время метод статического распределения на основе примеров почти всегда показывает худшие результаты. Такие результаты получаются вследствие того, что первый метод использует априорную информацию о количестве опорных векторов и, следовательно, о времени обучения для каждой рубрики. В то же время количество примеров для данной рубрики не всегда соответствует времени обучения.

Например, в табл. 1 приведены данные по нескольким рубрикам из рассматриваемой коллекции Reuters-21578.

Таблица 1. Данные по десяти рубрикам коллекции Reuters-21578 с наибольшим количеством примеров

Условные номера рубрик	Количество примеров в рубрике	Время обучении для рубрики, с	Количество опорных векторов
31	2877	8,53	766
1	1650	9,79	857
62	538	6,77	520
38	433	4,96	386
25	389	5,42	414
112	369	6,13	438
46	347	5,40	436
114	212	2,89	211
96	197	4,89	387
17	181	4,12	331

Из табл. 1 видно, что количество примеров в рубрике слабо связано с временем обучения, при этом время обучения сильно коррелирует с количеством опорных векторов.

Таким образом, в методе статического распределения на основе опорных векторов сбалансированность распределения нагрузки по узлам оказывается в среднем выше, чем у остальных двух методов, за счет априорной информации.

6. Заключение

Таким образом, можно сделать вывод, что предлагаемые методы распределения нагрузки между узлами вычислительного кластера можно применять для повышения эффективности решения задачи текстовой классификации. При этом для коллекции Reuters-21578 достигается максимальное ускорение 2,75, максимальная эффективность 0,42.

По результатам экспериментов можно рекомендовать использование для обучающих коллекций подобного уровня 9–11 узлов вычислительного кластера, дальнейшее увеличение ведет к падению ускорения.

В случае, когда априорная информация о количестве опорных векторов (времени обучения) для каждой рубрики неизвестна, могут применяться методы статического распределения нагрузки на основе примеров и динамического распределения, причем последнему следует отдавать предпочтение. При решении задачи поиска оптимальных параметров классификатора на первом проходе рекомендуется использовать метод динамического распределения, а при по-

следующих – метод статического распределения на основе опорных векторов, причем можно корректировать распределение рубрик на каждом шаге в зависимости от результатов (количества опорных векторов для рубрик) предыдущего прохода.

Литература

1. Пескишева Т.А. Современные системы и модули автоматической рубрикации текстовых документов: Вятский государственный гуманитарный университет. – Киров, 2010. – 37 с.: – Библиогр. 30 назв. – Рус. – Деп. в ВИНИТИ 01.07.2010, №410 – В 2010.
2. Vapnik V. Statistical learning theory. Wiley, New York, 1998.
3. Sebastiani F. Machine Learning in Automated Text Categorization. ACM Computing Surveys, Vol. 34, No. 1, March 2002, pp. 1–47.
4. Weston J., Watkins C. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, Egham, TW20 0EX, UK, 1998.
5. Bottou L., Cortes C., Denker J., Drucker H., Guyon I., Jackel L., Le Cun Y., Muller U., Sackinger E., Simard P., Vapnik V. Comparison of classifier methods: a case study in handwriting digit recognition. In International Conference on Pattern Recognition, pp. 77–87. IEEE Computer Press, 1994.
6. Krebel B. Pairwise classification and support vector machines. In B. Schölkopf, C. J. C. Burges, A. J. Smola, editors. Advances in Kernel Methods – Support Vector Learning, pp. 255–268, Cambridge, MA, 1999. MIT Press.
7. Platt J., Cristianini N., Shawe-Taylor J. Large Margin DAGS for Multiclass Classification. In Advances in Neural Information Processing Systems, 12 ed. S. A. Solla, T. K. Leen and K.-R. Muller, MIT Press, 2000. Pp. 547–553.
8. Котельников Е. В., Стародубова Т. А. Параллельные алгоритмы многоклассовой классификации на основе метода опорных векторов // Высокопроизводительные параллельные вычисления на кластерных системах. Материалы восьмой Международной конференции-семинара. – Казань: КГТУ, 2008, с. 228-233.
9. Project MPI.NET // The Open Systems Lab, Indiana University.

   URL: http://www.osl.iu.edu/research/mpi.net (дата обращения: 01.02.2011).
10. Reuters-21578, Distribution 1.0.

    URL: http://www.daviddlewis.com/resources/testcollections/reuters21578 (дата обращения: 01.02.2011).
11. Морфологический анализатор Mystem от компании Yandex.
    URL: http://company.yandex.ru/technology/mystem (дата обращения: 01.02.2011).
12. Salton G. Term-weighting approaches in automatic text retrieval // Information Processing & Management. – [s.l.] : Pergamon Press, 1988. – 5 : Vol. 24. – pp. 513-523.